The Dissertation Committee for Hyuk Park
certifies that this is the approved version of the following dissertation:

Truncated Multiplications and Divisions for the Negative Two's Complement Number System

Committee:

Earl E. Swartzlander, Jr., Supervisor

Anthony P. Ambler

Mircea D. Driga

William L. Gallagher

Nur A. Touba

Truncated Multiplications and Divisions for the Negative Two's Complement Number System

by

Hyuk Park, B.S., M.S.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

December 2007

This dissertation is dedicated to my parents,

Park, Yong Gon and Kim, Kyeong Ja.

Acknowledgments

I owe my gratitude to my supervisor, Dr. Earl E. Swartzlander, Jr. for the support, guideline and encouragement. Dr. Swartzlander is not only a dedicated supervisor, but also a role model whom I look up to. Without him, the research described in this dissertation would not have been possible.

I am also grateful to the members of my dissertation committee, Dr. Anthony P. Ambler, Dr. Mircea D. Driga, Dr. Nur A. Touba, and Dr. William L. Gallagher for their keen insights in my research.

The intellectual and mental support from all my colleagues at the Application Specific Processor Lab has provided me with a lot of help. Thank you. Finally, but not the least, I would like to thank my parents and my brothers for their love and support.

Truncated Multiplications and Divisions for the Negative Two's Complement Number System

Publication No. _____

Hyuk Park, Ph.D.
The University of Texas at Austin, 2007

Supervisor: Earl E. Swartzlander, Jr.

In the design of digital signal processing systems, where single-precision results are required, the power dissipation and area of parallel multipliers can be significantly reduced by truncating the less significant columns and compensating to produce an approximate rounded product. This dissertation presents the design of truncated multiplications of signed inputs utilizing a new number system, the negative fractional two's complement number system which solves an inherent problem of the conventional two's complement number system. This research also presents a new truncated multiplication method to reduce the errors with only slightly more hardware. Error, area, delay and dynamic power estimates are performed at the structural HDL level. The new method is also applied to various conventional number systems.

For division, which is the slowest and most complex of the arithmetic operations, a new truncated division method is described that yields the same

errors as those of true rounding without additional execution time that is normally required for true rounding. The new method is also applied to various conventional number systems.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

Recently minimizing power consumption has become essential in the design of digital signal processing systems due to the demand of mobile systems. In the design of many digital signal processing systems, the results from the basic arithmetic operations like addition, substraction, multiplication and division are maintained in a constant word length. In multiplication, which significantly occupies the overall power consumption of these systems, two inputs yield a product in doubled word length. Even though the desired word length is in single word length, the full addition of the partial product is executed before rounding to get accurate results since carries from the least significant partial product bits influence the rounded product. For these systems, truncated multiplication methods are introduced by [1–4] to save complexity and power consumption. These techniques do not form the partial product bits in the least significant columns, but they compensate for the not formed bits to get close to the desired precision.

In the previous works, truncated multiplication was applied in the unsigned number system and the two's complement number system. Unfortunately, the conventional fractional two's complement number system has an

inherent problem in that -1 multiplied by -1 yields -1 instead of $+1$. This problem also occurs in truncated multiplications. To solve this inherent problem, a new number system, the fractional negative two's complement number system has been introduced in [5]. This system is a mirror image of the conventional two's complement number system and contains $+1$ thus avoiding the inherent problem in the conventional two's complement number system.

In the design of many systems, it is desirable to reduce the execution time of division since it is the slowest of the basic arithmetic operations. Among algorithms for division, digit-recurrence division is widely used. Digit-recurrence division is performed by sequences of shifts and adds, generating each quotient digit by an iteration consisting of a shift and an add. To get a rounded quotient result, an extra iteration and addition for true rounding are required resulting in an increased execution time.

To mitigate the increase of the execution time, a constant round value is preset into the dividend in order to effect an increase in value of the final quotient so that when the quotient result is truncated (without the extra iteration and addition for true rounding) an approximately rounded result is obtained [6]. Even though this method reduced the execution time, the divisor needs to be normalized to get the approximately rounded result.

1.2 Research Direction

For multiplication, this research introduces signed truncated multiplication in a new number system, the negative two's complement number system, which avoids the extreme error inherent to two's complement multiplication, analyzes errors in truncated multiplication and, furthermore, proposes a new truncated multiplication method which yields smaller errors than the previous truncation methods. The new method is applied to not only the unsigned number system but also the conventional two's complement number system and the negative two's complement number system. The new truncated multiplication is compared with the previous works for the unsigned number system, the conventional two's complement number system and the negative two's complement number system in terms of error, dynamic power consumption, area and delay.

For division, this dissertation introduces truncated division in the floating point number system, the conventional two's complement number system and the negative two's complement number system. Furthermore, a new method for truncated division is proposed. The new method with a small hardware increase yields the same errors as those of true rounding without the extra iteration and addition for true rounding as well as normalization of the divisor and it is applied to the unsigned number system, the floating point number system, the two's complement number system and the negative two's complement number system. Errors of the previous methods and the new method are compared via exhaustive simulation.

1.3 Dissertation Organization

The organization of this dissertation is as follows. In Chapter 2, the background of truncated multiplication and truncated division is summarized and the previous researches are introduced. In Chapter 3, a new number system, the negative two's complement number system, which solves the inherent problem, is introduced and shown to be applicable to the fundamental arithmetic operations and associated operations. This is followed by truncated multiplication in the negative two's complement number system and examination of errors in signed truncated multiplication in Chapter 4. In Chapter 5, a new truncated multiplication is proposed for the unsigned number system, the conventional two's complement number system and the negative two's complement number system. This new method and the previous methods are compared by simulating in structural HDL design. In Chapter 6, truncated division is extended to the floating point number system, the conventional two's complement number system and the negative two's complement number system. In Chapter 7, a new truncated division is introduced and compared to true rounded division and the previous method. Chapter 8 summaries the research work of the dissertation and gives suggestions for future research.

Chapter 2

Previous Work

Accurate arithmetic costs full hardware usage and execution time. In the design of many systems, the result of arithmetic operations with reasonably small error is acceptable. Therefore, cost can be reduced by taking a result which has a reasonably small error instead of the maximally accurate result. This chapter reviews previous research on multiplication with reduced hardware and division with reduced execution time.

2.1 Truncated Multiplication

In digital signal processing systems, multiplication is often encountered [7]. Parallel multipliers provide fast multiplication, but need a large number of transistors, resulting in large power consumption and an expensive cost. To mitigate these problems, various techniques have been used including truncated multiplication [1–4]. These techniques also focus on reducing the errors that result from truncation. Therefore, the errors are small enough for many digital signal processing applications.

2.1.1 Multiplication with True Rounding

In multiplication, which significantly occupies the overall power consumption of these systems, two n bit inputs yield a product in 2n bit word length for unsigned numbers (or 2n-1 bit word length for two's complement numbers). Even though the desired word length is n bits, the full addition of the partial product is executed before rounding to get accurate results since carries from the least significant partial product bits influence to the rounded product in Figure 2.1.

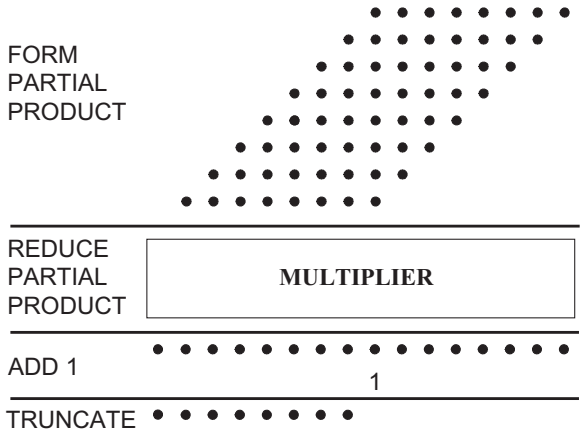

Figure 2.1: Rounded multiplication.

2.1.2 Truncated Multiplication with Correction

Figure 2.2 shows truncated multiplication with correction. To save complexity and reduce the power consumption of the multiplier, the partial product bits in the less significant columns are truncated. This process re-

sults in a large error since carries from the truncated partial product bits are not counted. This large error due to the truncated partial product bits is compensated by the error estimate to get the desired precision. Methods to compensate the error estimate have been introduced by [1–4]. The constant correction method [1–3] adds a constant to the remaining columns that is equal to the average value of the unformed bits. In the variable correction method [4], the partial product bits in the most significant column, which are eliminated in the constant correction method, are added in the rightmost column. These values are equivalent to multiplying them by 2. The variable correction method yields lower errors than the constant correction methods, but it has very slightly more complexity [8].

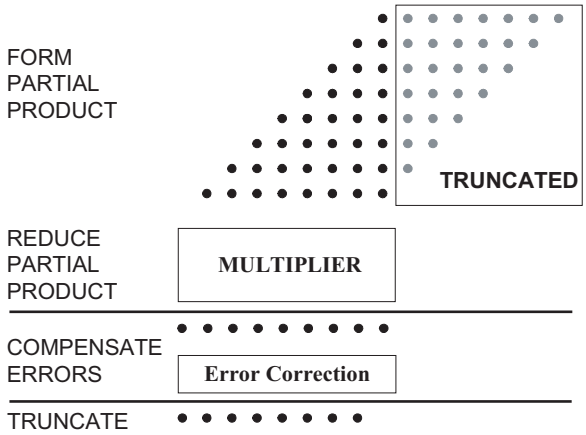

Figure 2.2: Truncated multiplication with correction.

2.1.2.1 Constant Correction Method

The multiplication of two n bit unsigned fractional numbers A and B yields a 2n bit product, P.

$$A = \sum_{i=0}^{n-1} a_i 2^{i-n} \qquad B = \sum_{i=0}^{n-1} b_i 2^{i-n} \qquad P = \sum_{i=0}^{2n-1} p_i 2^{i-2n}.$$

In Figure 2.3, the partial product bits in the n+k most significant columns are used for the product for truncated multiplication [9]. The average error by truncating the partial product bits in the n-k least significant columns and rounding the final result to n bits is estimated and compensated. Each bit of A and B is equally likely to be 0 or 1 if the operands are randomly chosen. Thus the each partial product has an average value of 0.25. For the product bits to be rounded, the average value of them is 0.5 since each is equally likely to be 0 or 1. By summing the average values of the truncated partial product bits and the product bits to be rounded, the average error is estimated by

$$E_{total} = -\sum_{q=0}^{n-k-1} \frac{1}{4}(q+1)2^{-2n+q} - 2^{-n-1}(1-2^{-k}) \text{ [2]}$$

For error compensation, E_{total} is rounded and added to the n-k column.

$$C = -\frac{round(2^{n+k}E_{total})}{2^{n+k}}$$

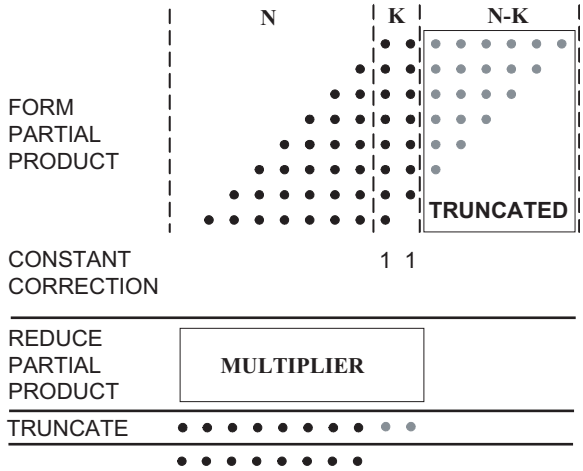

Figure 2.3: Truncated multiplication with constant correction method.

2.1.2.2 Variable Correction Method

In the truncated multiplication with constant correction, the maximum error occurs if the truncated partial product bits are all 0s or all 1s. When the truncated partial product bits are all 0s, the error associated with the final product is the correction constant. When the truncated partial product bits are all 1s, the correction constant is not big enough to compensate for the error due to the truncation.

In truncated multiplication with variable correction [4], the partial product bits in the n-k-1 column are added to the n-k column to compensate the truncated partial product bits in the n-k least significant columns in Figure 2.4. When the partial product bits in the n-k-1 column are all 0s, a correction value is 0. If the number of 1s in the n-k-1 equals the number of 0s, a correction value equals the correction constant in the constant correction

method. When the partial product bits in the n-k-1 column are all 1s, some maximum value compensates the final product. For an error by rounding the final result to n bits, the value of C_{round} is also added to the n-k column [9].

$$C_{round} = 2^{-n-1}(1 - 2^{-k+1})$$

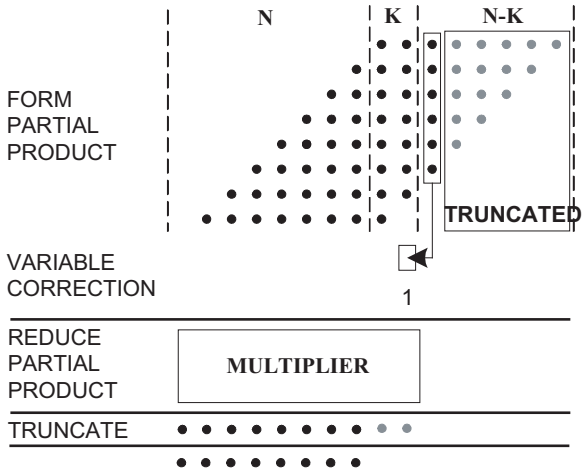

Figure 2.4: Truncated multiplication with variable correction method.

Both the constant correction method and the variable method significantly reduce the complexity of rounded multipliers with reasonably small errors. The constant correction method has slightly less hardware than the variable correction method. The variable correction method achieves smaller errors than the constant correction method [8].

2.2 Truncated Division

Of the basic arithmetic operations, division is the most complex and due to its inherently serial nature it is the hardest to speed up [10]. Among algorithms for division, digit-recurrence division is widely used [11]. In digit-recurrence division, one digit of the quotient is generated by each iteration of shifting and addition [12]. Digit-recurrence division is performed by sequences of shifts and adds. When the iterations are performed in series, the execution time is proportional to the number of digits of the quotient. For a rounded result, an additional iteration of shift and add is required. Furthermore carry propagate addition is also required for true rounding. To mitigate the increase of the execution time, a constant round value may be preset into the dividend in order to effect an increase in value of the final quotient so that when the quotient result is truncated (without the extra iteration and addition for true rounding) an approximately rounded result is obtained [6].

2.2.1 Division with True Rounding

Non-restoring division is faster than restoring division since it does not need to restore the partial remainder which is temporarily incorrect for some iterations during restoring division. In unsigned non-restoring division, digits of the quotient, Q, are selected as follows [13].

$$P_{k+1} = r \cdot P_k - q_{n-k-1} \cdot D \quad for\ k = 0, 1, ..., n-1$$

where P_k is the partial remainder after selecting the kth quotient digit, D is the divisor, $P_0 = N < D$, and r is the radix. When radix is 2, the quotient digits can be -1 or +1 and the digit for the kth iteration is selected by

$$If\ P_k \geq 0,\ q_{n-k-1} = +1\ and\ P_{k+1} = 2P_k - D$$
$$If\ P_k < 0,\ q_{n-k-1} = -1\ and\ P_{k+1} = 2P_k + D$$

After n iterations of shifts and adds, the n quotient bits are produced. In the correction step, the quotient is decreased by 1 to get a positive remainder when the remainder after n iterations is negative. For a rounded quotient, an additional iteration of shift and add is required to get a round bit. For true rounding, the round bit is added, resulting in carry propagate addition. This procedure is shown in Figure 2.5.

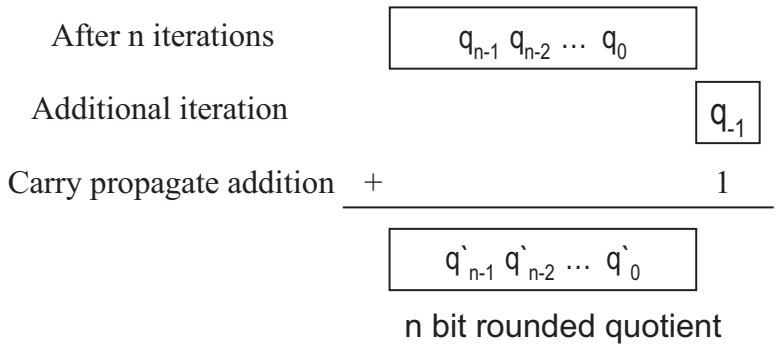

Figure 2.5: Additional procedure in division with true rounding.

2.2.2 Truncated Division with Constant Correction

In rounded division, the additional iteration and carry propagate addition steps require increasing the execution time. To avoid the increase in execution time, a constant round value, 1/3 ulp, is preset into the dividend in order to effect approximate rounding of the quotient [6]. After the quotient is calculated, it is truncated without the extra iteration and addition that are normally required for true rounding. When the divisor is normalized before division, the range of divisor is [1/2, 1) and the quotient value for the constant round value is in the range of (1/3, 2/3]. This quotient value for this case has errors bounded between -2/3 ulp and 2/3 ulp. Since 1/3 is represented by 010101... in binary, 0 and 1 are alternately preset into the dividend each iteration in Figure 2.6. Note that each bit of 1/3 ulp is added to the partial remainder before being divided by the divisor during each iteration. As the length of data is shorter, the constant round value is less than 1/3 yielding a slightly asymmetric error distribution.

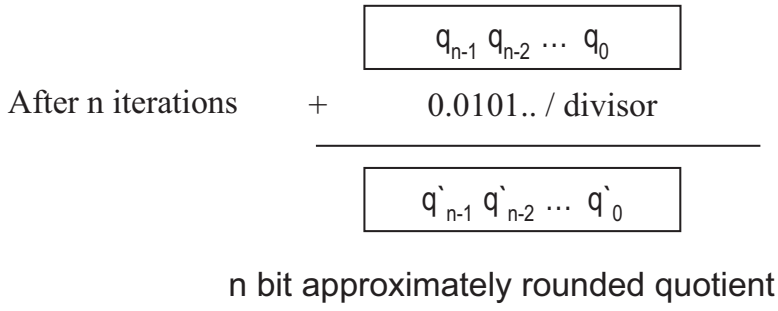

Figure 2.6: Division with the constant round value.

In unsigned binary non-restoring division, digits of the quotient, Q, are selected as follows [13].

$$P_{k+1} = 2P_k - q_{n-k-1} \cdot D \quad \textit{if } k \textit{ is even } for\ k = 0, 1, ..., n-1$$

$$P_{k+1} = 2P_k + 2^{-n} - q_{n-k-1} \cdot D \quad \textit{if } k \textit{ is odd } for\ k = 0, 1, ..., n-1$$

where P_k is the partial remainder after selecting the kth quotient digit and $P_0 = N < D$. Note that 0 and 1 are alternatively added to the LSB of the shifted dividend each iteration. Since the LSB of the shifted dividend is always 0 before presetting the constant constant value, it never requires a carry propagate addition. The quotient digits can be -1 or $+1$ and the digit for the kth iteration is selected by

$$\textit{If } P_k \geq 0,\ q_{n-k-1} = +\ 1 \textit{ and } P_{k+1} = 2P_k - D$$

$$\textit{If } P_k < 0,\ q_{n-k-1} = -\ 1 \textit{ and } P_{k+1} = 2P_k + D$$

After n iterations of shifts and adds, the n quotient bits, which are approximately rounded, are produced without an additional iteration and addition for rounding.

Chapter 3

The Negative Two's Complement Number System

This chapter provides an overview of a new number system, the fractional negative two's complement number system proposed in [5]. This number system is applicable to the fundamental arithmetic operations and it solves an inherent problem in the conventional two's complement number system.

3.1 Overview

Although floating point arithmetic is becoming widely used in DSP chips, many high performance systems use fixed point arithmetic [14]. In many applications, the fractional two's complement fixed point number system is commonly used [13], but the conventional fractional two's complement number system has an inherent problem in that -1 multiplied by -1 yields -1 instead of $+1$ since the number system is not capable of representing $+1$. To eliminate this problem, a new number system, the fractional negative two's complement number system has been introduced in [5]. Since this number system is a mirror image of the conventional two's complement number system, it avoids the problem inherent in the conventional two's complement number

system by including +1.

3.2 Definition

The formula for numbers in the new number system is identical to that of the conventional fractional two's complement number system except that the signs of all weights are reversed. An n bit fractional negative two's complement number, A, has a sign bit, a_{n-1}, and n-1 fractional bits, a_{n-2}, a_{n-3}, ..., a_0.

$$A = a_{n-1} - \sum_{i=0}^{n-2} a_i 2^{i-n+1}$$

Table 3.1 shows that the fractional negative two's complement number system is a mirror image of the fractional conventional two's complement number system.

3.3 Arithmetic Operations

Since the formula of fractional negative two's complement numbers is identical to that of fractional conventional two's complement numbers except for the reversed signs of all weights, the arithmetic operations of addition, subtraction and complementation are the same for both fractional negative two's complement numbers and conventional two's complement numbers.

Table 3.1: 4 bit fractional negative two's complement and conventional two's complement numbers

Value	Negative 2's complement	Conventional 2's complement
1	1000	N/A
7/8	1001	0111
3/4	1010	0110
5/8	1011	0101
1/2	1100	0100
3/8	1101	0011
1/4	1110	0010
1/8	1111	0001
0	0000	0000
-1/8	0001	1111
-1/4	0010	1110
-3/8	0011	1101
-1/2	0100	1100
-5/8	0101	1011
-3/4	0110	1010
-7/8	0111	1001
-1	N/A	1000

3.3.1 Addition and Subtraction

For fractional negative two's complement numbers, addition of two operands with a carry input yields the sum as for unsigned binary numbers without regard for the sign of the operands [15]. For subtraction of fractional negative two's complement numbers, the subtrahend is complemented by inverting all bits of it and adding 1 at the LSB before it is added to the minuend as with the conventional two's complement number system. Figure 3.1 shows examples of addition and subtraction for the negative two's complement number system.

	+1/2	1. 1 0 0		-1/4	0. 0 1 0
	+3/8	1. 1 0 1		+3/8	1. 1 0 1
=	+7/8	1. 0 0 1	=	+1/8	1. 1 1 1
	+1/2	1. 1 0 0		-1/4	0. 0 1 0
	-7/8	0. 1 1 1		-1/2	0. 1 0 0
=	-3/8	0. 0 1 1	=	-3/4	0. 1 1 0

Figure 3.1: Examples of addition and subtraction for the negative two's complement number system.

3.3.2 Overflow

As for the conventional two's complement number system [15], detection of overflow can be performed by the exclusive-OR of the carry into the

sign bit and the carry out from the sign bit [16]. Figure 3.2 shows examples of overflow for the negative two's complement number system. In the left case, the carry in is 0 and the carry out is 1. In the right case, the carry in is 1 and the carry out is 0.

	+1/2	1. 1 0 0		-1/2	0. 1 0 0
	+7/8	1. 0 0 1		-3/4	0. 1 1 0
=	-5/8	0. 1 0 1	=	+3/4	1. 0 1 0

Figure 3.2: Examples of overflow for the negative two's complement number system.

If the word size of the result above is increased by 1 bit, overflow can be accommodated. The extended bit has a weight of 2 and the next significant bit has a weight of -1 instead of $+1$. Note that it results in growth of the result range. Figure 3.3 shows accommodation of overflow in Figure 3.2.

	+1/2	1. 1 0 0		-1/2	0. 1 0 0
	+7/8	1. 0 0 1		-3/4	0. 1 1 0
=	+1 3/8	1 0. 1 0 1	=	-1 1/4	0 1. 0 1 0

Figure 3.3: Examples of overflow accommodation for the negative two's complement number system.

3.3.3 Multiplication

For multiplication of two operands, two types of multipliers are widely used: Booth/modified Booth multipliers and parallel multipliers. Differently from addition and subtraction, implementation of multipliers for negative two's complement numbers has some difference from that for conventional two's complement numbers.

3.3.3.1 Booth Multiplier

Both Booth [17] and modified Booth multipliers [18] are performed with sequences of additions, subtractions and shifts. Since the operations in the sequence are the same for the conventional two's complement number system and the negative two's complement number system, the resulting multipliers for both number system are similar. The only difference in negative two's complement Booth multipliers is shown in Table 3.2, in which addition and subtraction conditions are reversed from those of conventional multipliers. Note that P is the partial product which is produced by adding the multiplier B, subtracting it or bypassing it on each cycle.

3.3.3.2 Parallel Multiplier

In fractional negative two's complement parallel multipliers, all the partial product bits are inverted and correction bits are added that are different from those of the modified Baugh and Wooley method used for two's complement numbers [10]. The multiplication of two n bit data for fractional negative

Table 3.2: Booth multiplier operations for negative two's compelement numbers.

Multiplier bits	Operation
0 0	P = P
0 1	P = P − B
1 0	P = P + B
1 1	p = p

two's complement numbers A and B yields a 2n-1 bit product, P.

$$A = a_{n-1} - \sum_{i=0}^{n-2} a_i 2^{i-n+1}$$

$$B = b_{n-1} - \sum_{i=0}^{n-2} b_i 2^{i-n+1}$$

$$P = p_{2n-2} - \sum_{i=0}^{2n-3} p_i 2^{i-2n+2}$$

Figure 3.4 shows the bit product matrix for multiplication in the fractional negative two's complement number system. The bit product $a_x b_y$ indicates a_x AND b_y, the bit product $\overline{a_x b_y}$ represents a_x NAND b_y, and 1 means a logic ONE. The addition of n 1s at the least significant portion in the last row does not significantly impact the delay or the complexity of the multiplier.

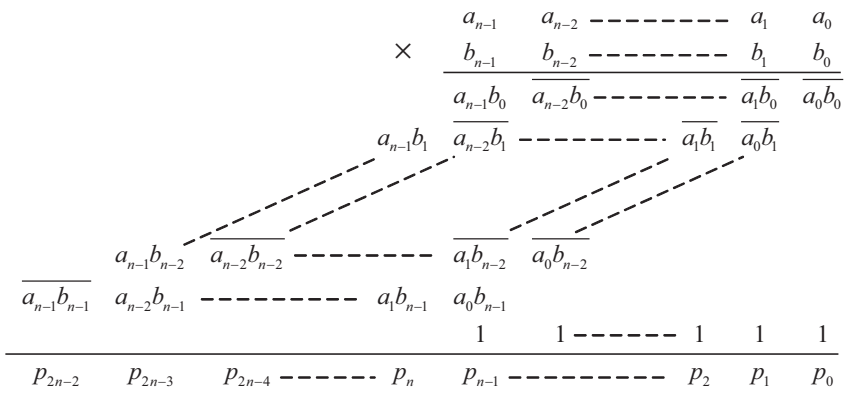

Figure 3.4: The bit product matrix for multiplication in the fractional negative two's complement number system.

3.3.4 Division

For division of two operands, two types of dividers are widely used: digit recurrence dividers [11] and fast dividers (e.g., Newton-Raphson dividers [19] and Goldschmidt dividers [20]). The former dividers are performed by sequences of shifts and adds with comparisons. When the quotient bits are in signed digit format, they needs to be converted into normal binary format. The latter dividers involve add and multiply operations. For dividers of negative two's complement numbers, comparison and conversion of the signed digit quotients into fractional negative two's complement numbers are described since all operations except them have been already described.

3.3.4.1 Comparison

To compare negative two's complement numbers, the sign bit is converted before the numbers are compared. Then the numbers will be in inverse lexicographic order, meaning that the smallest is all 1s and the largest is all 0s. Therefore, after normal comparison of the numbers with the inverse sign bit, the result is inverted.

3.3.4.2 Conversion from Signed Digit to Negative Two's Complement

For conventional two's complement numbers, signed digit numbers can be converted into two's complement numbers by separating the signed digit number into a number denoted by P with 0 and positive digits, and a number denoted by N with 0 and negative digits. Then calculation of P-N yields the conversion [10]. The MSB of the result is sign extended.

For negative two's complement numbers, the same process is applied except that N-P is calculated at the end of the process instead of P-N. Figure 3.5 shows examples of conversion from a radix 2 signed digit number to negative two's complement number.

3.3.4.3 Conversion between Negative Two'Complement and Conventional Two's Complement Numbers

Since the two systems are mirror images of each other, conversion from one to the other is performed by inverting all bits and adding 1 to the LSB, which is same process for the complementation of the number. Since +1 in

SD =	$.1\,0\,\bar{1}\,0\,1\,1\,0\,\bar{1}\,\bar{1}\,0$	= 426/1024
P =	$.1\,0\,0\,0\,1\,1\,0\,0\,0\,0$	
- P =	$.0\,1\,1\,1\,0\,1\,0\,0\,0\,0$	
N =	$.0\,0\,1\,0\,0\,0\,0\,1\,1\,0$	
N - P =	$1.1\,0\,0\,1\,0\,1\,0\,1\,1\,0$	= 426/1024

SD =	$.\bar{1}\,0\,1\,0\,\bar{1}\,\bar{1}\,0\,1\,1\,0$	= -426/1024
P =	$.0\,0\,1\,0\,0\,0\,0\,1\,1\,0$	
- P =	$.1\,1\,0\,1\,1\,1\,1\,0\,1\,0$	
N =	$.1\,0\,0\,0\,1\,1\,0\,0\,0\,0$	
N - P =	$0.0\,1\,1\,0\,1\,0\,1\,0\,1\,0$	= -426/1024

Figure 3.5: Examples of conversion from signed digit number to negative two's complement number.

the negative two's complement number system and -1 in the conventional two's complement number system have no corresponding number to the other system, an attempt to convert either of them yields an incorrect result.

3.4 Summary

In this chapter, the fractional negative two's complement number system, which was proposed in [5], was reviewed. This number system solves the inherent problem of the conventional two's complement number system in that -1 multiplied by -1 yields -1 instead of $+1$. This number system is also applicable to the fundamental arithmetic operations of addition, subtraction, multiplication and division as well as the associated operations of complementation, conversion from/to conventional two's complement numbers and signed digit numbers.

Chapter 4

Truncated Multiplication

In the unsigned number system, truncated multiplication with the constant correction method has been shown in [1–3] and truncated multiplication with the variable correction method has been shown in [4]. This chapter analyzes problems of signed truncated multiplications in the conventional two's complement number system. Another extreme error in the constant correction method has been found by software simulation. Then truncated multiplication of signed inputs in a new number system, the negative fractional two's complement number system [5], is proposed for both the constant correction method and the variable correction method. Finally, errors of truncated multiplication with the constant correction method and the variable correction method for unsigned numbers, conventional two's complement numbers and negative two's complement numbers are compared by a software simulation with all possible inputs.

4.1 Overview

In digital signal processing systems, multiplication is often encountered [7]. Parallel multipliers [21–26] provide fast multiplication, but need a large

number of transistors, resulting in large power consumption and an expensive cost. To mitigate these problems, various techniques have been used including truncated multiplication [1–4]. These techniques also focus on reducing the errors that result from truncation. Therefore, the errors are small enough for many digital signal processing applications.

Although floating point arithmetic is widely used in DSP chips, high performance systems almost always use fixed point arithmetic [14]. In fixed point arithmetic, the unsigned number system with separated sign and the two's complement number system are usually chosen. In many applications, the fractional two's complement fixed point number system is commonly used [13], but the conventional fractional two's complement number system has an inherent problem in that -1 multiplied by -1 yields -1 instead of $+1$. Figure 4.1 shows an example of the inherent problem in the conventional fractional two's complement number system.

$$
\begin{array}{rcccccccccccc}
-1 & & & & & & 1 & 0 & 0 & 0 & 0 & 0 & \\
\times -1 & & & & & & 1 & 0 & 0 & 0 & 0 & 0 & \\
\hline
 & & & & & & 1 & 0 & 0 & 0 & 0 & 0 & \\
 & & & & & 1 & 0 & 0 & 0 & 0 & 0 & & \\
 & & & & 1 & 0 & 0 & 0 & 0 & 0 & & & \\
 & & & 1 & 0 & 0 & 0 & 0 & 0 & & & & \\
 & & 1 & 0 & 0 & 0 & 0 & 0 & & & & & \\
 & 1 & 1 & 1 & 1 & 1 & 1 & & & & & & \\
\hline
 & & & & & & 1 & & & & & & \\
\hline
 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & = -1
\end{array}
$$

Figure 4.1: 6-bit conventional two's complement multiplication with two -1 inputs [10, 27].

To solve this problem, a new number system, the negative two's complement number system [5] has been introduced. As reviewed in the previous chapter, the new system can express values for $1 \geq V > -1$. The product of any two numbers is also in the range $1 \geq P > -1$. The formula for numbers in the new number system is identical to that of the conventional fractional two's complement number system except that the signs of all weights are reversed.

4.2 Truncated Multiplications for Unsigned Fractions

In the truncated multiplication of two n bit unsigned fractional numbers, n least significant bits are truncated from the 2n bit product to maintain constant word size, n. To compensate for errors from truncation, the constant correction method [1–3] and the variable correction method [4] have been introduced. These methods utilize the n+k most significant columns from the partial product bits and provide an approximation for the other columns. For the unsigned fractional number system, both correction methods yield relatively small errors.

4.3 Truncated Multiplications for Two's Complement Fractions

For the truncated multiplication of two n bit two's complement fractional numbers, a constant word size, n, is preserved from 2n-1 bit product by

truncating the n-1 least significant bits. In both the constant and variable correction methods, the n+k+1 most significant columns of the partial product bits are manipulated and an approximation for the other columns is computed and added to compensate for the data that are truncated.

4.3.1 Inherent Errors

For the conventional two's complement number system, both of the correction methods suffer from an inherent error problem for a special case, in which multiplication of -1 and -1 yields -1 or slightly greater than -1, as in full multiplication. Although a correction constant is added to the partial product bits before truncation in the constant correction method, it is not enough to get the desired result, $+1$. Therefore, the truncated result is always -1 or a little bigger than -1. Similarly, for the variable correction method, because of no correction carry for the multiplication of -1 and -1, the truncated result is always -1. Figure 4.2 and Figure 4.3 show the extreme error examples of the constant and the variable correction method for the conventional two's complement system.

4.3.2 Extreme Errors in the Constant Correction Method

The constant correction truncation method for the conventional two's complement number system has another extreme error. When -1 is multiplied by the smallest negative value above -1, the correct result is the biggest pos-

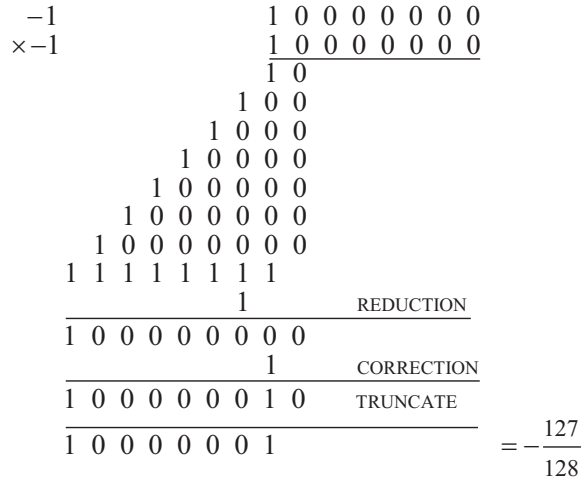

Figure 4.2: Extreme error of constant correction truncation for the conventional two's complement system with k=1.

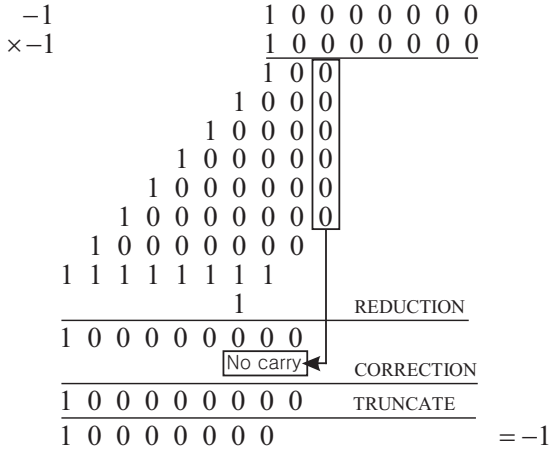

Figure 4.3: Extreme error of variable correction truncation for the conventional two's complement system with k=1.

itive value. The result after truncation with the constant correction method can be -1 if the correction constant is big enough to cause carry propagation. Note that all the truncated bit products are 0s in this case. Figure 4.4 shows this error situation. If the size of k is big enough, this error doesn't happen since the correction constant is too small to cause carry propagation.

$$
\begin{array}{rl}
-\frac{127}{128} & 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1 \\
\times -1 & \underline{1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0} \\
 & 1\ 0 \\
 & 1\ 0\ 0 \\
 & 1\ 0\ 0\ 0 \\
 & 1\ 0\ 0\ 0\ 0 \\
 & 1\ 0\ 0\ 0\ 0\ 0 \\
 & 1\ 0\ 0\ 0\ 0\ 0\ 0 \\
 & 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\
 & 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0 \\
 & \underline{\qquad\qquad\qquad\qquad 1 \qquad\qquad\qquad\qquad} \quad \text{REDUCTION} \\
 & 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0 \\
 & \underline{\qquad\qquad\qquad\qquad\qquad\qquad 1 \qquad\qquad\qquad} \quad \text{CORRECTION} \\
 & 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \quad \text{TRUNCATE} \\
 & \underline{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad} \\
 & 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \qquad\qquad\qquad\qquad = -1
\end{array}
$$

Figure 4.4: Another extreme error of constant correction truncation for the conventional two's complement system with k=1.

4.4 Truncated Multiplications for Negative Two's Complement Fractions

For the negative two's complement number system, which solves the inherent problem above, the constant correction method [1–3] and the variable correction method [4] are modified since the bit product matrix of the multi-

plication of the negative two's complement numbers is different from that of conventional two's complement numbers.

4.4.1 The Constant Correction Method

Figure 4.5 shows the bit product matrix for the constant correction method for the negative two's complement number system. Note that logical ONEs and a correction constant, $c_{n+k-1}, c_{n+k-2}, \ldots, c_0$, can be summed.

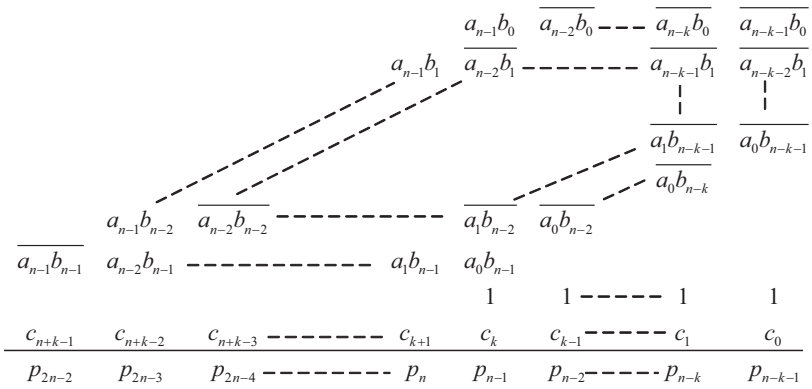

Figure 4.5: The bit product matrix in the constant correction method for the negative two's complement number system.

To get the correction constant, which is used to compensate for the truncation error, errors are examined.

The reduction error is,

$$E_{reduct} = \sum_{q=0}^{n-k-2} \left(\frac{3}{4}(q+1) + 1 \right) 2^{-2n+q+2}$$

The rounding error is,

$$E_{round} = \frac{1}{2} \sum_{q=n-k-1}^{n-2} 2^{-2n+q+2} = 2^{-n}(1 - 2^{-k})$$

Thus, the total error is,

$$E_{total} = \sum_{q=0}^{n-k-2} \left(\frac{3}{4}(q+1) + 1 \right) 2^{-2n+q+2} + 2^{-n}(1 - 2^{-k})$$

For correct correction,

$$C = -\frac{round(2^{n+k-1} E_{total})}{2^{n+k-1}}$$

4.4.1.1 Extreme Errors in the Constant Correction Method

Although the multiplication of the negative two's complement numbers has no inherent error, truncation with constant correction has some limitations. The negative two's complement number system can have an extreme error when both of the inputs are one. If the absolute value of correction constant is smaller than the absolute value of the partial product bits not formed, the

truncated result is a little bigger than -1 instead of equal to $+1$ since the bit product matrix has not enough carries to produce the desired final product bits, in which the MSB is 1 and all the others are 0s. Full multiplication of two 8-bit negative two's complement numbers with inputs of 1s in Figure 4.6 shows the result of 1, but its truncated multiplication with small size of k in Figure 4.7 shows the extreme error.

$$
\begin{array}{rcccccccccccccccc}
1 & & & & & & & & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\
\times 1 & & & & & & & & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\
\hline
 & & & & & & & & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \\
 & & & & & & & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & & \\
 & & & & & & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & & & \\
 & & & & & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & & & & \\
 & & & & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & & & & & \\
 & & & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & & & & & & \\
 & & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & & & & & & & \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & & & & & & & \\
\hline
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & =1
\end{array}
$$

Figure 4.6: Multiplication of two 8-bit negative two's complement numbers with full precision.

The extreme error above does not always happen. If the value of k is big enough, the error doesn't occur. The equation below insures that no extreme error will occur in constant correction truncation for the negative two's complement number system. Table 7.1 generated with this equation shows the minimum sizes of k for each input size, n, to avoid the extreme error.

$$|correction\ constant| \geq (n-k-1)2^{-n-k+1}$$

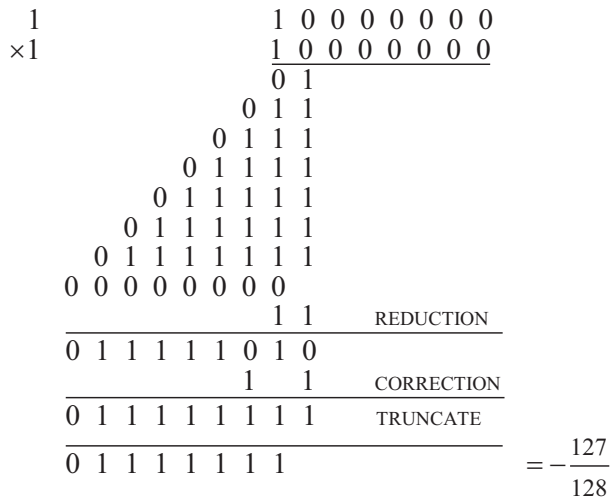

Figure 4.7: Multiplication of two 8-bit negative two's complement numbers with the constant correction method with k=1.

Table 4.1: Minimum size of k for each input size without an extreme error

Minimum k for no extreme error	Input size (n)
1	2 - 6
2	7 - 11
3	12 - 20
4	21 - 37
5	38 - 71
6	72 - 136
7	137 - 265

4.4.2 The Variable Correction Method

In the variable correction method for multiplication in the negative two's complement number system, there is no extreme error, like that which can occur with the constant correction method. In this method, the partial product bits of the n-k-2 column including the logical ONE, which compensates for the logical ONEs of the columns not formed, are carried into the n-k-1 column in Figure 4.8 (a). This logical ONE carried into the n-k-1 column results in just a single logical ONE in the n column after the sum of logical ONEs at the bottom of partial product bits in Figure 4.8 (b). Logical ONEs on the top of partial product bits represents a rounding constant to compensate for the rounding error [9]. The value of the rounding constant in the negative two's complement number system is

$$C_{round} = -2^{-n}(1 - 2^{-k+1})$$

When one of inputs is 0 or 1, which has all zeros except the MSB, the partial product bits of the n-1 least significant columns are all 1s since a NAND with one of inputs as zero always outputs one. In this case, the sum of partial product bits not formed and partial product bits of the n-k-2 column is the exactly twice the value of the partial product bits of the n-k-2 column except logical ONE. The value of logical ONE carried from the n-k-2 column is canceled after truncation. It means that the carries from the partial product bits in the n-k-2 column to the n-k-1 column, help to yield exactly the same

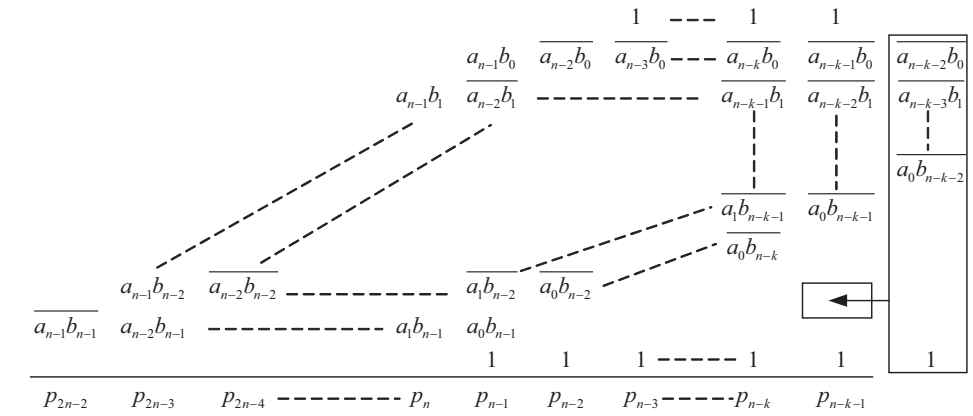

Figure 4.8: (a) The bit product matrix in the variable correction method for the negative two's complement number system, (b) The modified bit product matrix in the variable correction method for the negative two's complement number system.

value as the correct result, avoiding the extreme error, which occurs with the constant correction method. Figure 4.9 represents this procedure.

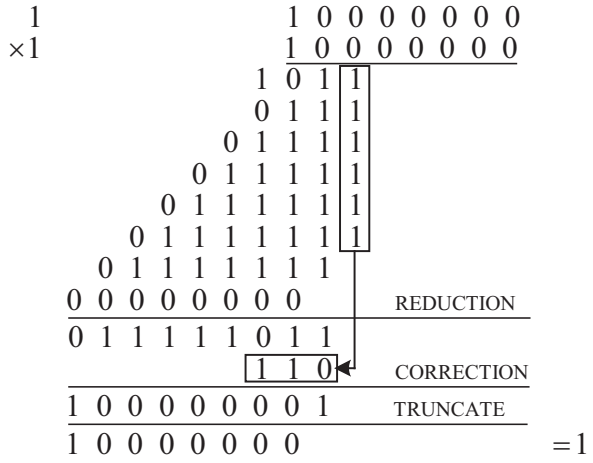

Figure 4.9: Example of the multiplication of two 8-bit negative two's complement numbers with the variable correction method with k=1.

4.5 Error Comparison

Table 7.2 shows the error comparison [28] of the constant correction truncation method and the variable correction truncation method for 7 x 7 unsigned numbers, 8 x 8 conventional two's complement numbers and 8 x 8 negative two's complement numbers for all possible input values. Note that the n-1 x n-1 unsigned, the n x n conventional two's complement and the negative two's complement number systems have the same number (n-k-1) of reduction columns. The correction methods were simulated with all possible inputs in MATLAB [29, 30] and C [31, 32]. The variance in the constant cor-

rection method for 8 x 8 conventional two's complement numbers is apparently higher than others because of three extreme errors, 1 inherent error and 2 carry propagation errors.

Table 4.2: Error comparison of constant correction truncation and variable correction truncation for 7 x 7 unsigned numbers, 8 x 8 conventional two's complement numbers and negative two's complement numbers

k	Number System	Max. positive error		Max. negative error		Mean		variance	
		CC	VC	CC	VC	CC	VC	CC	VC
1	Unsigned	1.000	0.695	-2.008	-0.945	0.125	-0.125	0.231	0.102
	Positive 2's	1.000	0.695	**-255**	**-256**	0.113	-0.129	3.210	1.101
	Negative 2's	1.000	0.695	**-255**	-0.945	0.121	-0.125	1.224	0.102
2	Unsigned	0.750	0.555	-1.008	-0.680	0.119	-0.064	0.109	0.086
	Positive 2's	0.750	0.555	**-256**	**-256**	0.115	-0.068	1.110	1.086
	Negative 2's	0.750	0.555	-1.008	-0.680	0.119	-0.064	0.109	0.086
3	Unsigned	0.500	0.508	-0.758	-0.570	-0.031	-0.035	0.088	0.083
	Positive 2's	0.500	0.508	**-256**	**-256**	-0.035	-0.039	1.087	1.083
	Negative 2's	0.500	0.508	-0.758	-0.570	-0.031	-0.035	0.088	0.083
4	Unsigned	0.500	0.492	-0.570	-0.523	0.002	-0.021	0.084	0.083
	Positive 2's	0.500	0.492	**-256**	**-256**	-0.002	-0.025	1.084	1.083
	Negative 2's	0.500	0.492	-0.570	-0.523	0.002	-0.021	0.084	0.083
5	Unsigned	0.500	0.492	-0.508	-0.508	0.012	-0.016	0.083	0.083
	Positive 2's	0.500	0.492	**-256**	**-256**	0.008	-0.018	1.083	1.083
	Negative 2's	0.500	0.492	-0.508	-0.508	0.012	-0.016	0.083	0.083
6	Unsigned	0.500	0.492	-0.492	-0.500	0.014	-0.014	0.083	0.083
	Positive 2's	0.500	0.492	**-256**	**-256**	0.010	-0.018	1.083	1.083
	Negative 2's	0.500	0.492	-0.492	-0.500	0.014	-0.014	0.083	0.083

If k is big enough for no extreme error in the multiplication of negative two's complement numbers, the n by n negative two's complement multiplication and the n-1 by n-1 unsigned multiplication have the same mean and maximum errors and similar variations in constant correction truncation since

the n by n negative two's complement multiplication except MSBs of two inputs and the n-1 by n-1 unsigned multiplication have the same error patterns and MSBs of two inputs of the n by n negative two's complement multiplication have 4 cases (00, 01, 10 and 11) with the same error patterns. This is more likely applied to variable correction truncation since it doesn't have any extreme error. This can be also applied to n by n conventional two's complement multiplication if it does not have extreme errors.

To compare the errors of the constant correction method and the variable correction method for 7 x 7 unsigned numbers, 8 x 8 conventional two's complement numbers and 8 x 8 negative two's complement numbers except extreme errors, software checking can be used to trap out input patterns which will produce extreme errors. In the conventional two's complement number system, the inherent extreme error always occurs for both the constant correction method and the variable correction method when the inputs are -1. Other extreme errors are from -1 multiplied by the smallest negative value above -1 for the constant correction method. In the negative two's complement number system, only the constant correction method has the extreme error when the inputs are 1s and k is not big enough. The error comparison of the various methods with the extreme cases omitted is shown in Table 4.3. In both the constant correction method and the variable correction method, the same maximum positive and negative errors and similar means and variances result for all three of the number systems.

Table 4.3: Error comparison of constant correction truncation and variable correction truncation for 7 x 7 unsigned numbers, 8 x 8 conventional two's complement numbers and negative two's complement numbers, omitting extreme errors

k	Number System	Max. positive error		Max. negative error		Mean		variance	
		CC	VC	CC	VC	CC	VC	CC	VC
1	Unsigned	1.000	0.695	-2.008	-0.945	0.125	-0.125	0.231	0.102
	Positive 2's	1.000	0.695	-2.008	-0.945	0.125	-0.125	0.231	0.102
	Negative 2's	1.000	0.695	-2.008	-0.945	0.125	-0.125	0.231	0.102
2	Unsigned	0.750	0.555	-1.008	-0.680	0.119	-0.064	0.109	0.086
	Positive 2's	0.750	0.555	-1.008	-0.680	0.119	-0.064	0.109	0.086
	Negative 2's	0.750	0.555	-1.008	-0.680	0.119	-0.064	0.109	0.086
3	Unsigned	0.500	0.508	-0.758	-0.570	-0.031	-0.035	0.088	0.083
	Positive 2's	0.500	0.508	-0.758	-0.570	-0.031	-0.035	0.088	0.083
	Negative 2's	0.500	0.508	-0.758	-0.570	-0.031	-0.035	0.088	0.083
4	Unsigned	0.500	0.492	-0.570	-0.523	0.002	-0.021	0.084	0.083
	Positive 2's	0.500	0.492	-0.570	-0.523	0.002	-0.021	0.084	0.083
	Negative 2's	0.500	0.492	-0.570	-0.523	0.002	-0.021	0.084	0.083
5	Unsigned	0.500	0.492	-0.508	-0.508	0.012	-0.016	0.083	0.083
	Positive 2's	0.500	0.492	-0.508	-0.508	0.012	-0.016	0.083	0.083
	Negative 2's	0.500	0.492	-0.508	-0.508	0.012	-0.016	0.083	0.083
6	Unsigned	0.500	0.492	-0.492	-0.500	0.014	-0.014	0.083	0.083
	Positive 2's	0.500	0.492	-0.492	-0.500	0.014	-0.014	0.083	0.083
	Negative 2's	0.500	0.492	-0.492	-0.500	0.014	-0.014	0.083	0.083

4.6 Summary

In the conventional two's complement number system, both truncated multiplication with the constant correction method and truncated multiplication with the variable correction method suffer from the inherent error in that -1 multiplied by -1 yields -1. To solve this problem, both correction methods for truncated multiplication in the negative two's complement number system have been proposed. By a software simulation, another extreme error in the constant correction method was found and the number of truncated columns to avoid the extreme errors is determined. If all extreme errors are omitted, both the constant correction method and the variable correction method have the same maximum positive and negative errors and similar means and variances for all three of the number systems.

Chapter 5

Truncated Multiplication with Nearly Correctly Rounding

In this chapter, a new correction method for truncated multiplication is proposed to reduce the errors with only slightly more complexity than the variable correction method. The new correction method can be applied to the unsigned number system, the conventional two's complement number system and the negative two's complement number system.

For demonstration of the new method and comparison with the constant correction method and the variable correction method, 8 bit multipliers and 16 bit multipliers for the unsigned number system, the conventional two's complement number system and the negative two's complement number system are designed in structural HDL, which are automatically generated by perl scripts, and synthesized by Synopsys Design Compiler in standard cells for TSMC 0.25 μm technology. The area, speed and dynamic power of different designs are compared.

5.1 Overview

In truncated multiplication, the constant correction method [1–3] and the variable correction method [4] have been used to improve the accuracy of truncated multipliers. The partial product bits, which are formed by these correction methods, are usually computed by parallel multiplication [21–25]. The variable correction method has slightly more complexity than the constant correction method, but it has lower errors [8]. Even though the variable correction method compensates errors efficiently, the number of columns below the LSB of the result, k, should be increased to reduce the maximum error, increasing hardware usage and power consumption, if it is slightly over the expected error limit like one unit in the last place.

5.2 Motivation

In truncated multiplications, partial product bits in the n-k least significant columns are not formed to achieve a significant savings of complexity and power consumption. To compensate for errors by this hardware omission, the variable correction method adds the partial product bits in the n-k-1 column to the n-k column and sets round constants, logical 1s, from the n-2 column to the n-k column. In the unsigned fractional number system, maximum errors

of the variable correction method are

$$|E_{variable}| \;=\; 0.5 + \sum_{i=1}^{\lfloor (n-k)/2 \rfloor} (n-k+2-2i) \cdot 2^{-k-2i-1} \qquad [33]$$

These errors occur when two input pattern and the k bit product, from the n-1 column to the n-k column, are

$$a_{n-k-1} \; ... \; a_0, \; b_{n-k-1} \; ... \; b_0 = \begin{cases} 0 \; x \; \overline{x} \; ... \; x \; \overline{x} \; 1, \; 0 \; x \; \overline{x} \; ... \; x \; \overline{x} \; 1 & \text{if n-k is even} \\ 0 \; x \; \overline{x} \; ... \; \overline{x} \; x \; 1, \; 0 \; \overline{x} \; x \; ... \; x \; \overline{x} \; 1 & \text{if n-k is odd} \end{cases}$$

$$p_{n-1} \; ... \; p_{n-k} = 1 \; ... \; 1$$

where x is 0 or 1. In these input patterns, the partial product bits of the n-k-1 column, which are carried into the n-k column for error compensation, are always all ZEROs. Therefore, the partial product bits in the n-k-1 least significant columns, which are not formed for hardware saving, make the maximum error with the k bit truncated product from the n-1 column to the n-k column, which is formed for the carry to the n most significant result. The maximum error of the k bit final product from the n-1 column to the n-k column is a half of unit in the last place of the n bit result since the rounding constants are added from the n-2 column to the n-k column.

5.3 New Truncated Multiplication for Unsigned Fractions

By utilizing the input patterns which yield the maximum errors in the variable correction method, the maximum error, mean error and variance can be reduced. A new correction method including a new logic with 4 inputs, $a_{n-k-1},\ b_{n-k-1},\ a_0,\ and\ b_0$, is proposed in Figure 5.1. The special logic in Figure 5.2, which is equivalent to $a_0 b_0 \overline{(a_{n-k-1} + b_{n-k-1})}$, is added to the n-k column as a partial product bit. It makes a mandatory carry to the the n-k column, when the partial product in the n-k-1 column are all ZEROs, the left off partial product bits are $\sum_{i=1}^{\lfloor (n-k)/2 \rfloor} (n-k+2-2i) \cdot 2^{-k-2i-1}$ and the k bit final product from the n-1 column to the n-k column is all 1s, meaning that the real value of the product without the rounding constant is 1 for the n-1 position and ZEROs for others.

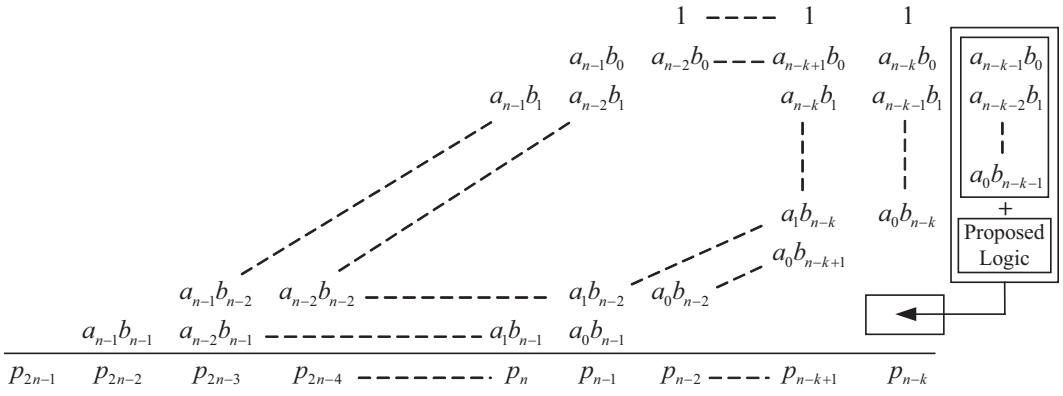

Figure 5.1: The bit product matrix for the new truncated multiplication method in the unsigned number system.

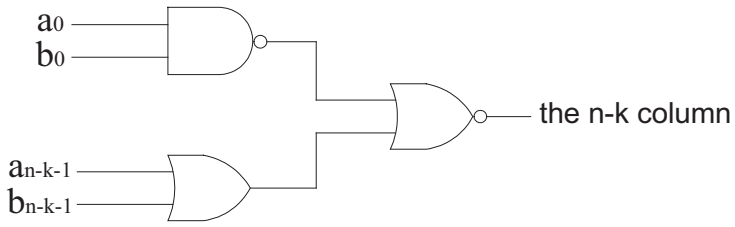

Figure 5.2: Special logic for the new truncated multiplication method in the unsigned number system.

5.3.1 Hardware Saving with a Specialized Counter

Even though the special logic has only about the same complexity as a half adder, the new correction method introduces one more partial product bit to the n-k column. It may increase the hardware for the partial product reduction. To mitigate this hardware increase, a specialized counter for the partial product bits including the special logic and two partial product bits is proposed. The partial product bits for error compensation to the n-k column consist of $a_{n-k-1}b_0$, $a_{n-k-2}b_1$ ... a_0b_{n-k-1}, and the special logic, $a_0b_0\overline{(a_{n-k-1}+b_{n-k-1})}$. If n-k is greater than 1, $a_{n-k-1}b_0$, a_0b_{n-k-1}, and the special logic always make a place for error compensation. Counting these three inputs shows an interesting property. Table 5.1 shows these three partial product bits never have 1s concurrently. This means that counting 4 inputs, which include $a_{n-k-1}b_0$, a_0b_{n-k-1}, and the special logic, yields just one carry and one sum as outputs for the partial product reduction. Table 5.2 shows an example of this benefit. When k is bigger than 1, the round constant 1 is always present in the n-k column. The logic equations for the carry and the sum of the spe-

Table 5.1: Truth table of the partial product bits for error compensation in the unsigned number system

a_0	b_0	a_{n-k-1}	b_{n-k-1}	$a_{n-k-1}b_0$	a_0b_{n-k-1}	$a_0b_0\overline{(a_{n-k-1}+b_{n-k-1})}$
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	0	1	0	0	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	0
1	0	0	1	0	1	0
1	0	1	0	0	0	0
1	0	1	1	0	1	0
1	1	0	0	0	0	1
1	1	0	1	0	1	0
1	1	1	0	1	0	0
1	1	1	1	1	1	0

cialized (4,2) counter with 4 inputs, $a_{n-k-1}b_0$ ($input1$), a_0b_{n-k-1} ($input2$), the special logic (input3) and round constant 1 (embedded input), are

$$Carry = input1 + input2 + input3$$

$$Sum = input1 \cdot (\overline{input2 \ \oplus \ input3})$$

which are simpler than those of a full adder. Since the specialized (4,2) counter yields only one carry and one sum, the new truncation method does not increase the number of counters regardless of the increase of one partial product bit in the n-k column. Only 6 gates are increased since the specialized (4,2) counter has just 3 gates more than a full adder and the special logic consists of 3 gates regardless of the size of input and k.

Table 5.2: Truth table of a specialized (4:2) counter in the unsigned number system

1	$a_{n-k-1}b_0$	a_0b_{n-k-1}	$a_0b_0\overline{(a_{n-k-1}+b_{n-k-1})}$	Sum	Carry
1	0	0	0	1	0
1	0	0	1	0	1
1	0	1	0	0	1
1	0	1	1	N/A	N/A
1	1	0	0	0	1
1	1	0	1	N/A	N/A
1	1	1	0	1	1
1	1	1	1	N/A	N/A

5.3.2 Truncated Array Multipliers with Unsigned Fractions

Figure 5.3 shows the block diagram of 6-bit unsigned truncated array multiplier with constant correction. The AHA cell consists of a half adder and an AND gate. The output of the AND gate is a partial product and is one input to the half adder. Similarly, the AFA cell consists of a full adder and an AND gate. In the least significant position of the bottom row, the SHA cell, which has comparable hardware and delay as that of a normal half adder, takes two inputs from the previous cells and computes the sum and carry of the two inputs plus 1 [23]. The ARHA cell is same to the AHA cell except that the output is only carry out. The outputs of the truncated array multiplier are numbered p_{11}, p_{10}, p_9, p_8, p_7, p_6 to match Figure 2.1. They may be renumbered p_5, p_4, p_3, p_2, p_1, p_0 to reflect their nature as a single precision 6-bit number.

The block diagram of 6-bit unsigned truncated array multiplier with variable correction is shown in Figure 5.4. Half adders along the upper right edge of the constant correction method are replaced by full adders due to the additional partial product bits. The ARFA cell is same to the AFA cell except that the output is only carry out.

Figure 5.5 shows the block diagram of 6-bit unsigned truncated array multiplier with the new correction method. The ARFA cell in the variable correction method is replaced by a specialized (4:2) counter with only carry out for the output in Figure 5.6. The SL cell has been shown in Figure 5.2.

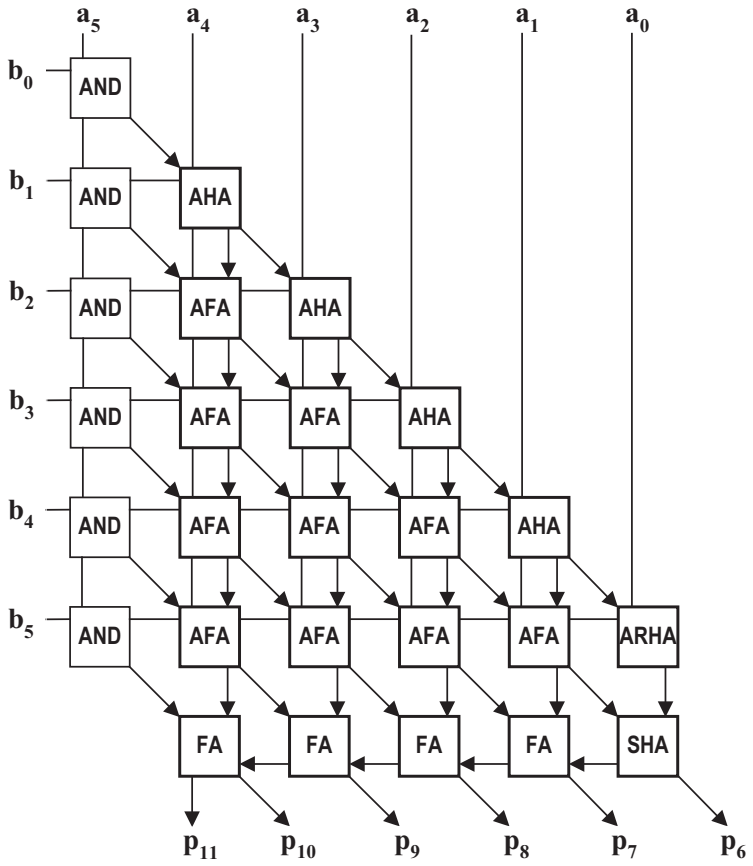

Figure 5.3: 6-bit unsigned truncated array multiplier with constant correction.

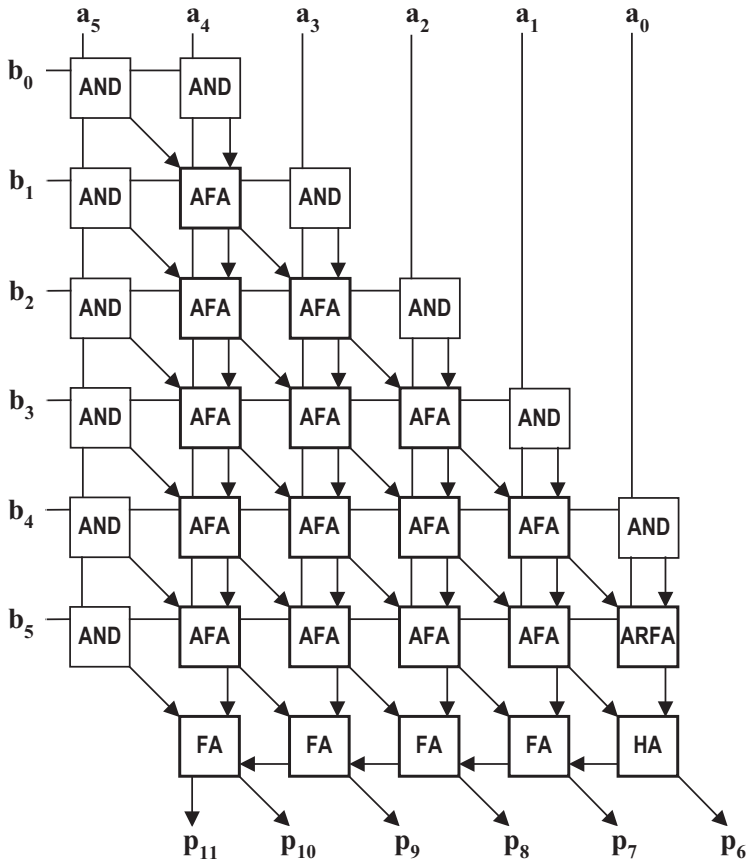

Figure 5.4: 6-bit unsigned truncated array multiplier with variable correction.

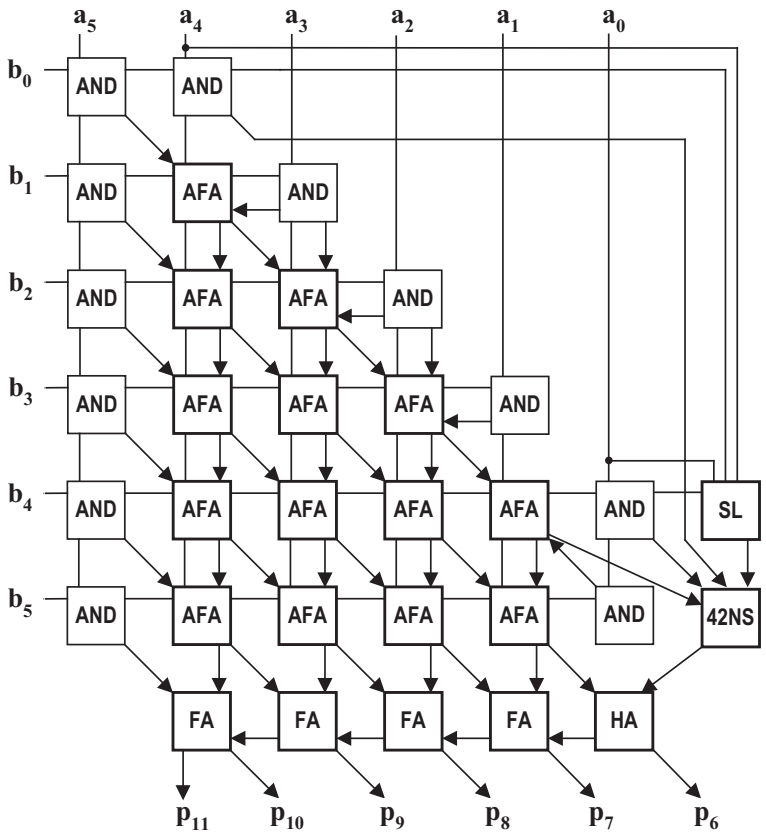

Figure 5.5: 6-bit unsigned truncated array multiplier with the new correction method.

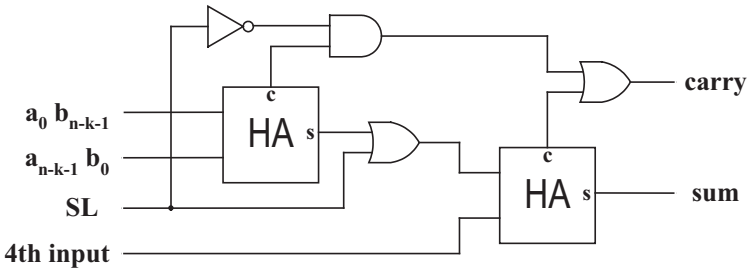

Figure 5.6: Specialized 4:2 counter for unsigned array multiplier.

5.3.3 Error Analysis

The error of the new correction method in units in the last place is

$$|E_{new}| \ \leq \ 0.5 + \sum_{i=1}^{\lfloor (n-k-1)/2 \rfloor} (n-k+1-2i) \cdot 2^{-k-2i-1}$$

which always has lower maximum absolute error than that of the variable correction method.

Table 5.3 shows the error comparison of the variable correction truncation method and the new correction truncation method for all possible input values in the unsigned number system for various sizes of n and k. The positive maximum errors of the two truncation methods are the same. The negative maximum error, which has larger absolute value than the positive maximum error, is reduced by the new correction method. The difference of the maximum error from the variable correction method to the new truncation method is

$$|D_{maximum}| = \sum_{i=1}^{\lfloor (n-k-2)/2 \rfloor} 2^{-k-2i-1} + 2^{-n}$$

This shows that maximum error of the new correction method is reduced when the size of k is smaller.

When n=8, k=1 and n=15, k=2, the maximum error of the variable correction truncation method is over 1 ulp [33] and that of the new correction

truncation method is under 1 ulp. This means that, in the variable correction truncation method, the size of k can be increased to make the maximum error less than 1 ulp, at a cost of increasing the complexity and power consumption. The mean error and variance of the new correction truncation is always less than those of the variable correction truncation even though Table 5.3 may indicate same mean error and variance because of data length. When k is 1, mean errors of two truncation method are fixed as -0.125 and -0.09375. As k is increased, the difference of the mean errors of the two truncation methods gets smaller.

Table 5.3: Error comparison of variable correction truncation and new correction truncation for all possible input values in the unsigned number system.

n	k	Max. positive error		Max. negative error		Mean		variance	
		VC	New	VC	New	VC	New	VC	New
6	1	0.609	0.609	-0.859	-0.781	-0.125	-0.094	0.096	0.095
	2	0.516	0.516	-0.641	-0.594	-0.066	-0.051	0.085	0.084
	3	0.484	0.484	-0.547	-0.531	-0.039	-0.031	0.083	0.083
7	1	0.695	0.695	-0.945	-0.859	-0.125	-0.094	0.102	0.101
	2	0.555	0.555	-0.680	-0.641	-0.065	-0.049	0.086	0.086
	3	0.508	0.508	-0.570	-0.547	-0.035	-0.027	0.084	0.084
8	1	0.777	0.777	**-1.027**	**-0.945**	-0.125	-0.094	0.107	0.106
	2	0.598	0.598	-0.723	-0.680	-0.063	-0.048	0.088	0.088
	3	0.527	0.527	-0.590	-0.570	-0.033	-0.025	0.084	0.084
10	1	0.944	0.944	-1.194	-1.111	-0.125	-0.094	0.118	0.117
	2	0.681	0.681	-0.806	-0.764	-0.063	-0.047	0.091	0.090
	3	0.569	0.569	-0.632	-0.611	-0.032	-0.024	0.085	0.085
12	1	1.111	1.111	-1.361	-1.278	-0.125	-0.094	0.128	0.127
	2	0.764	0.764	-0.889	-0.847	-0.063	-0.047	0.093	0.093
	3	0.611	0.611	-0.674	-0.653	-0.031	-0.024	0.085	0.085
15	1	1.361	1.361	-1.611	-1.528	-0.125	-0.094	0.144	0.143
	2	0.888	0.888	**-1.014**	**-0.972**	-0.063	-0.047	0.097	0.097
	3	0.674	0.674	-0.736	-0.715	-0.031	-0.023	0.086	0.086

Error histograms of 8-bit unsigned truncated multiplication with the the constant correction method, the variable correction method and the new method by MATLAB simulation are shown in Figure 5.7, Figure 5.8 and Figure 5.9. Errors of the constant correction method are biased and the absolute value of the negative maximum error is greater than 2.5. Even though errors of the variable correction method are less biased, more negative errors are distributed and the absolute value of the negative maximum error is greater than 1. The new method shows the most symmetric error distribution. Furthermore, the absolute value of the negative maximum error is less than 1 and the positive maximum error equals that of the variable correction method.

5.4 New Truncated Multiplication for Two's Complement Fractions

The new correction method presented in the previous sections is extended for the conventional fractional two's complement number system with an inherent error problem for a special case, in which multiplication of -1 and -1 yields -1. If the inherent error is ignored, the new correction method is similarly applied to the conventional two's complement number system as is to the unsigned number system. The n-k-1 least significant columns are not formed instead of the n-k least significant columns before error compensation. The partial product bits in the n-k-2 column and the new partial product bit are added to the n-k-1 columns for error compensation. Figure 5.10 shows the new truncation multiplication method for the conventional two's complement

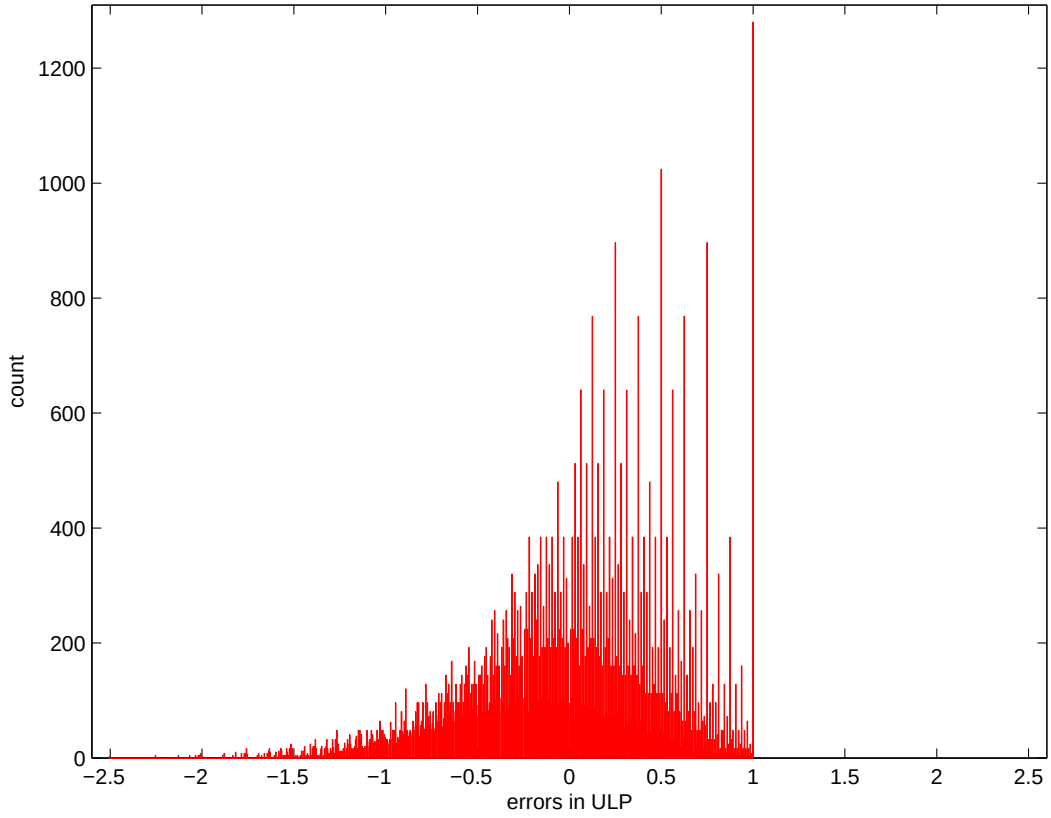

Figure 5.7: Error histogram of 8-bit unsigned truncated multiplication with the constant correction method with k = 1.

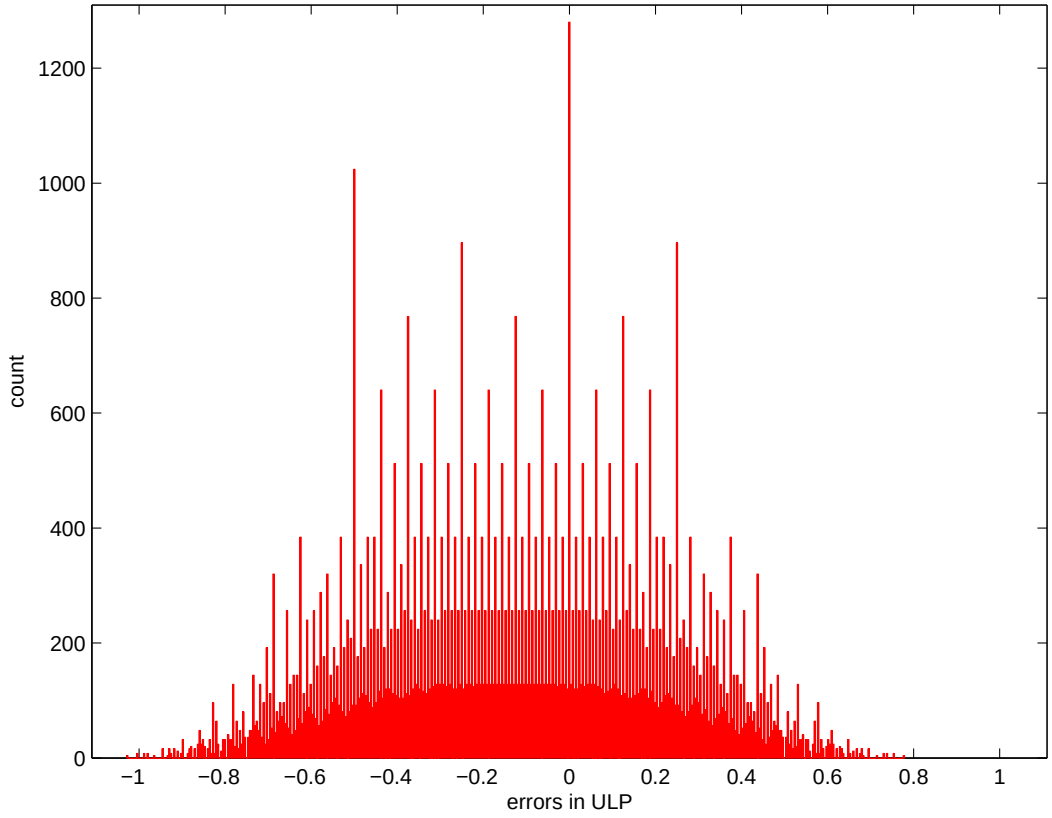

Figure 5.8: Error histogram of 8-bit unsigned truncated multiplication with the variable correction method with k = 1.

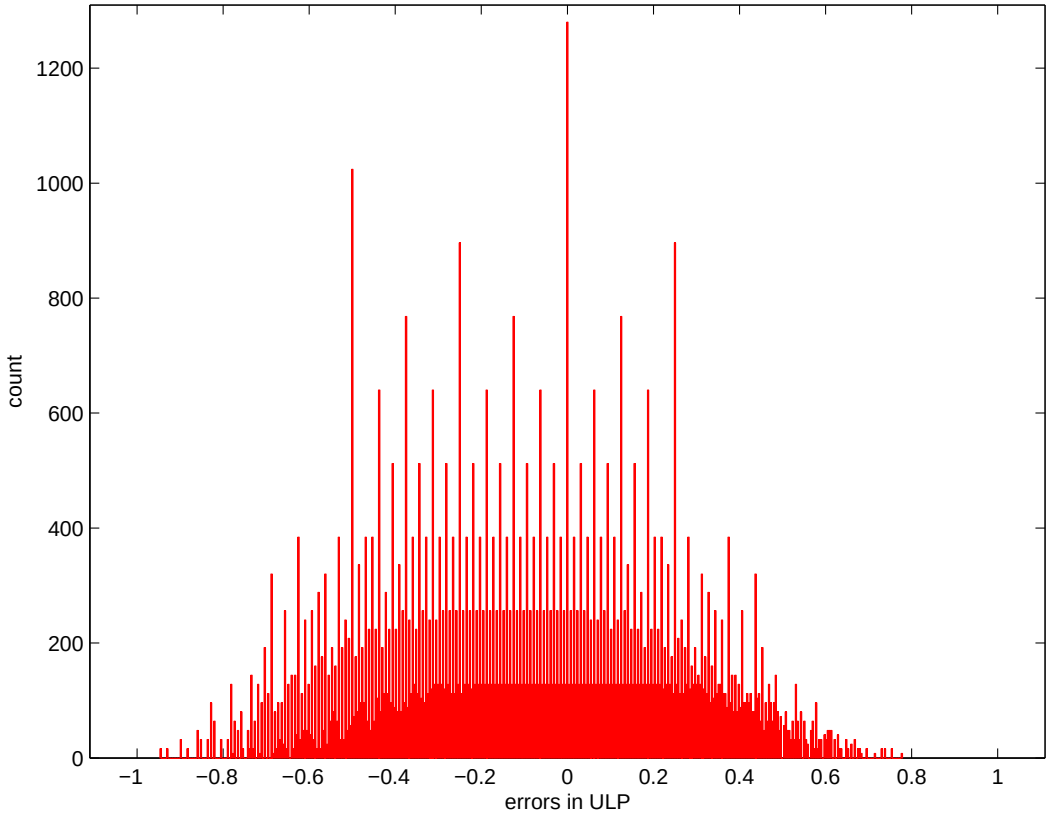

Figure 5.9: Error histogram of 8-bit unsigned truncated multiplication with the new method with k = 1.

number system with the special logic in Figure 5.11. Inputs to the special logic are $a_{n-k-2},\ b_{n-k-2},\ a_0,\ and\ b_0$ instead of $a_{n-k-1},\ b_{n-k-1},\ a_0,\ and\ b_0$.

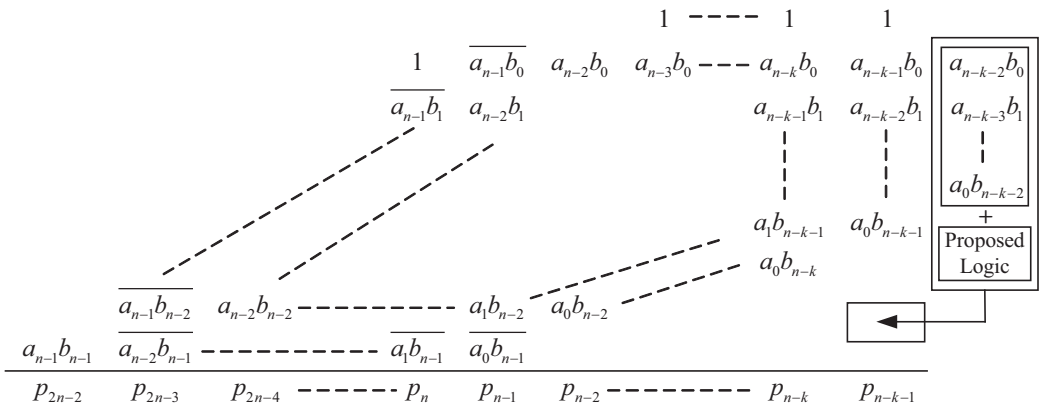

Figure 5.10: The bit product matrix for the new truncated multiplication method in the conventional two's complement number system.

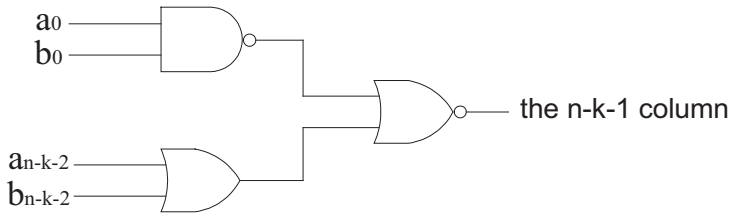

Figure 5.11: Special logic for the new truncated multiplication method in the conventional two's complement number system.

5.4.1 Hardware Saving with a Specialized Counter

As in the unsigned number system, a specialized (4:2) counter is also available in the conventional two's complement number system to mitigate hardware increase by one more partial product bit in the n-k-1 column. Since

error compensation bits, $a_{n-k-2}b_0,\ a_0b_{n-k-2}$, and $a_0b_0\overline{(a_{n-k-2}+b_{n-k-2})}$ never have 1s at the same time, the count of 4 inputs including these inputs never exceeds 3, meaning only one carry and one sum are needed. As in the unsigned number system, if embedded input4 is 1, a round constant, the logic equations for the carry and the sum of the specialized (4,2) counter are

$$Carry = input1 + input2 + input3$$

$$Sum = input1 \cdot (\overline{input2 \ \oplus \ input3})$$

5.4.2 Truncated Array Multipliers with Two's Complement Fractions

Figure 5.12 shows the block diagram of 6-bit two's complement truncated array multiplier with constant correction. The NFA cell consists of a full adder and an NAND gate. The FANC cell is a full adder with no carry out.

The block diagram of 6-bit two's complement truncated array multiplier with variable correction is shown in Figure 5.13. Half adders along the upper right edge of the constant correction method are replaced by full adders due to the additional partial product bits.

Figure 5.14 shows the block diagram of 6-bit two's complement truncated array multiplier with the new correction method. As in the unsigned number system, the ARFA cell in the variable correction method is replaced by a specialized (4:2) counter with only carry out for the output in Figure 5.15.

The SL cell has been shown in Figure 5.11.

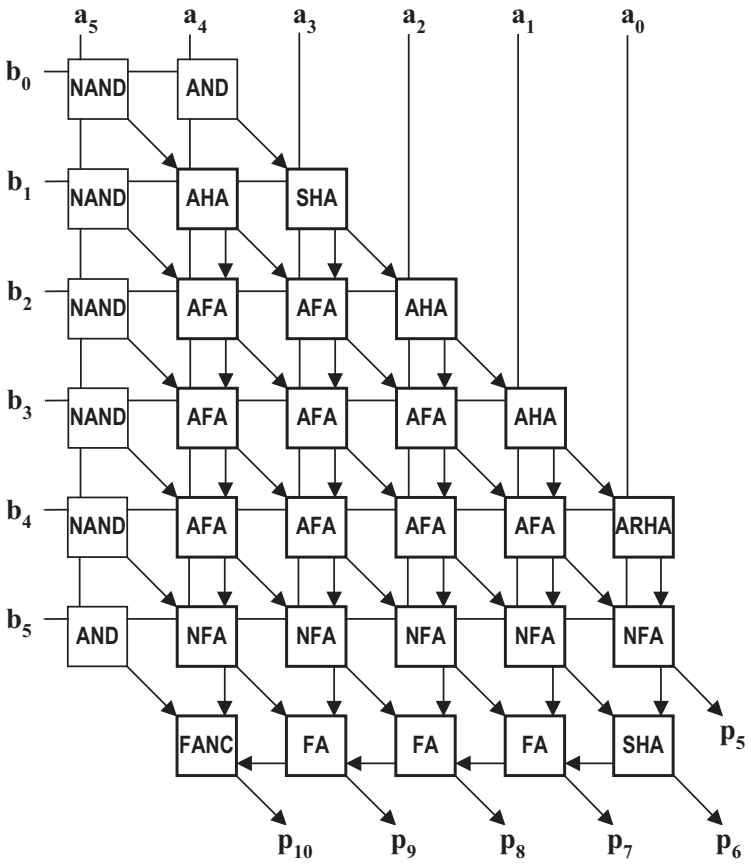

Figure 5.12: 6-bit two's complement truncated array multiplier with constant correction.

5.4.3 Error Analysis

If the inherent error is not considered, errors of the variable correction method in units in the last place for the conventional two's complement number

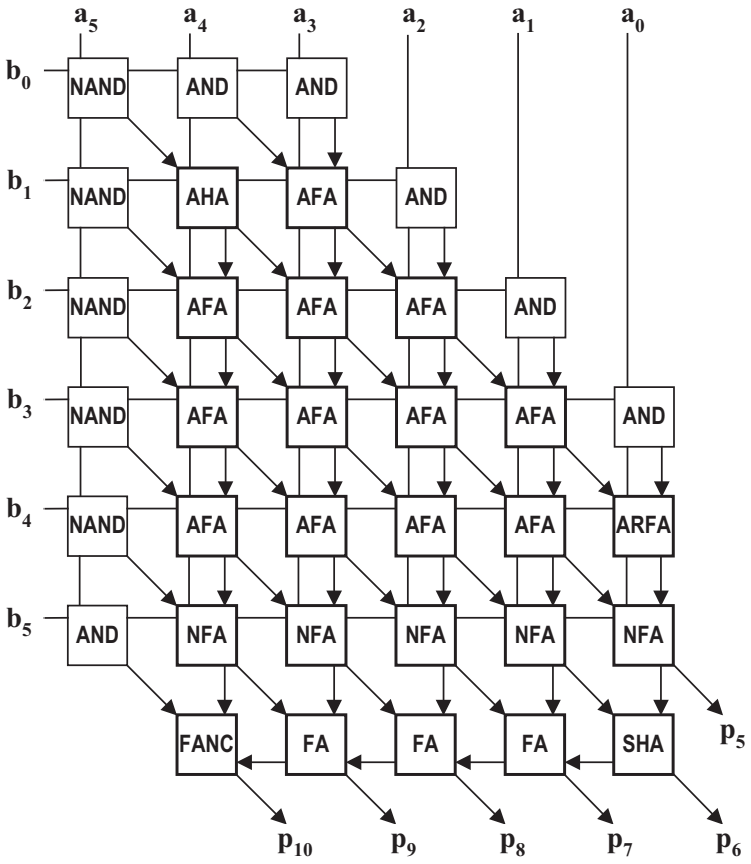

Figure 5.13: 6-bit two's complement truncated array multiplier with variable correction.

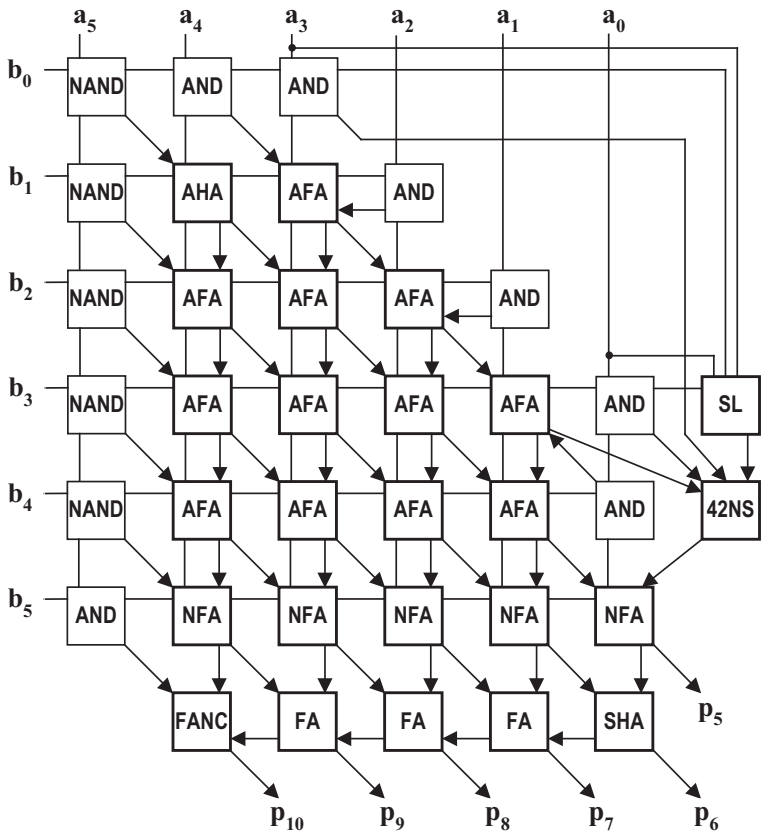

Figure 5.14: 6-bit two's complement truncated array multiplier with the new correction method.

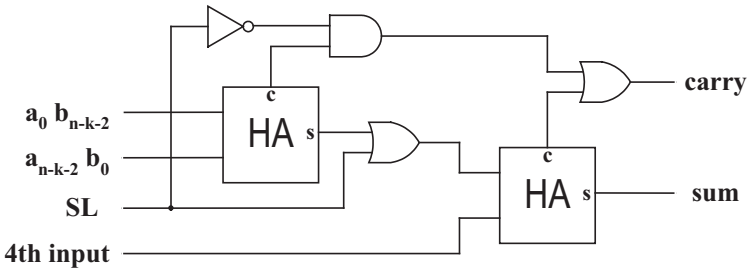

Figure 5.15: Specialized 4:2 counter for two's complement array multiplier.

system is

$$|E_{variable}| \ \leq \ 0.5 + \sum_{i=1}^{\lfloor (n-k-1)/2 \rfloor} (n-k+1-2i) \cdot 2^{-k-2i-1}, \ if \ A = B \neq 1$$

The error of the new correction method in units in the last place is

$$|E_{new}| \ \leq \ 0.5 + \sum_{i=1}^{\lfloor (n-k-2)/2 \rfloor} (n-k-2i) \cdot 2^{-k-2i-1}, \ if \ A = B \neq 1$$

Table 5.4 shows the error comparison of variable correction truncation and the new correction truncation method for all possible input values, except when both inputs are -1, in the conventional two's complement number system for various sizes of n and k. The new correction method reduces the maximum error by

$$|D_{maximum}| = \sum_{i=1}^{\lfloor (n-k-3)/2 \rfloor} 2^{-k-2i-1} + 2^{-n+1}, \ if \ A = B \neq 1$$

Error histograms of 8-bit two's complement truncated multiplication with the the constant correction method, the variable correction method and the new method are shown in Figure 5.16, Figure 5.17 and Figure 5.18. Errors of the constant correction method are biased and the absolute value of the negative maximum error is greater than 2. Even though errors of the variable correction method are less biased and the absolute value of the negative

Table 5.4: Error comparison of variable correction truncation and new correction truncation for all possible input values, except -1 by -1, in the conventional two's complement number system.

n	k	Max. positive error		Max. negative error		Mean		variance	
		VC	New	VC	New	VC	New	VC	New
7	1	0.609	0.609	-0.859	-0.781	-0.125	-0.094	0.096	0.095
	2	0.516	0.516	-0.641	-0.594	-0.066	-0.051	0.085	0.084
	3	0.484	0.484	-0.547	-0.531	-0.039	-0.031	0.083	0.083
8	1	0.695	0.695	-0.945	-0.859	-0.125	-0.094	0.102	0.101
	2	0.555	0.555	-0.680	-0.641	-0.065	-0.049	0.086	0.086
	3	0.508	0.508	-0.570	-0.547	-0.035	-0.027	0.084	0.084
9	1	0.777	0.777	**-1.027**	**-0.945**	-0.125	-0.094	0.107	0.106
	2	0.598	0.598	-0.723	-0.680	-0.063	-0.048	0.088	0.088
	3	0.527	0.527	-0.590	-0.570	-0.033	-0.025	0.084	0.084
11	1	0.944	0.944	-1.194	-1.111	-0.125	-0.094	0.118	0.117
	2	0.681	0.681	-0.806	-0.764	-0.063	-0.047	0.091	0.090
	3	0.569	0.569	-0.632	-0.611	-0.032	-0.024	0.085	0.085
13	1	1.111	1.111	-1.361	-1.278	-0.125	-0.094	0.128	0.127
	2	0.764	0.764	-0.889	-0.847	-0.063	-0.047	0.093	0.093
	3	0.611	0.611	-0.674	-0.653	-0.031	-0.024	0.085	0.085
16	1	1.361	1.361	-1.611	-1.528	-0.125	-0.094	0.144	0.143
	2	0.888	0.888	**-1.014**	**-0.972**	-0.063	-0.047	0.097	0.097
	3	0.674	0.674	-0.736	-0.715	-0.031	-0.023	0.086	0.086

maximum error is less than 1, more negative errors are distributed. By applying the new method, errors are more symmetric than for the other methods. The absolute value of the negative maximum error is also decreased and the positive maximum error equals that of the variable correction method.

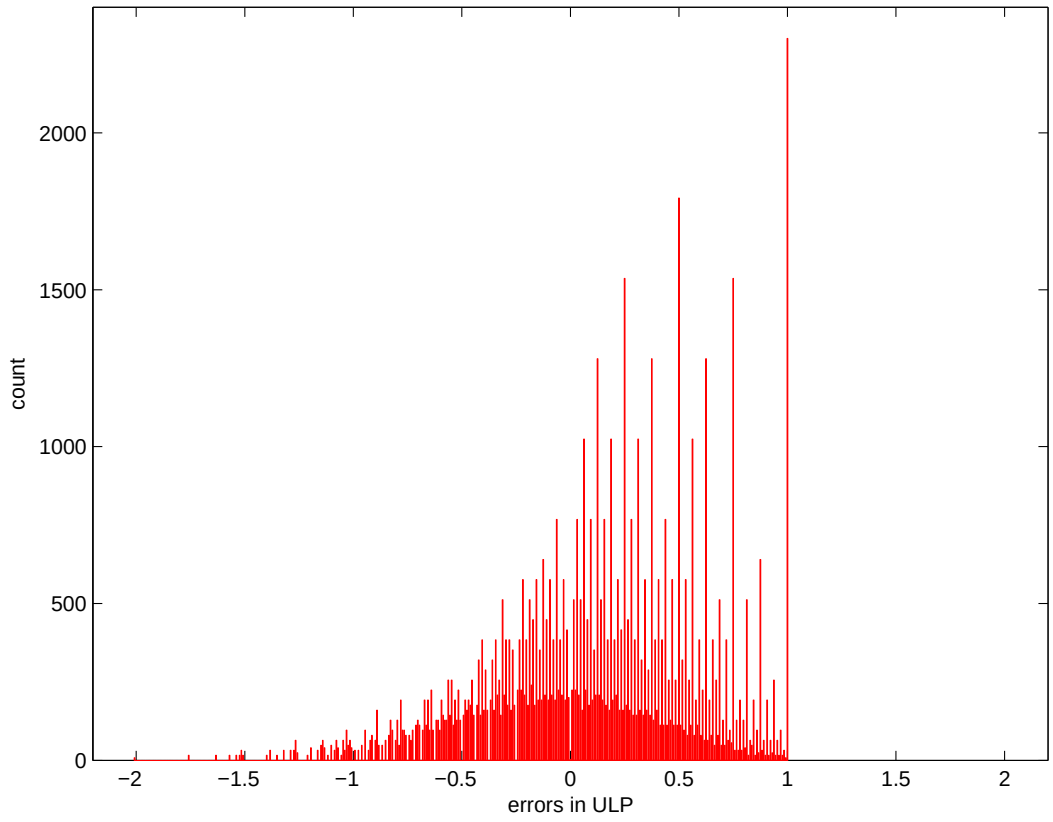

Figure 5.16: Error histogram of 8-bit two's complement truncated multiplication with the constant correction method with k = 1.

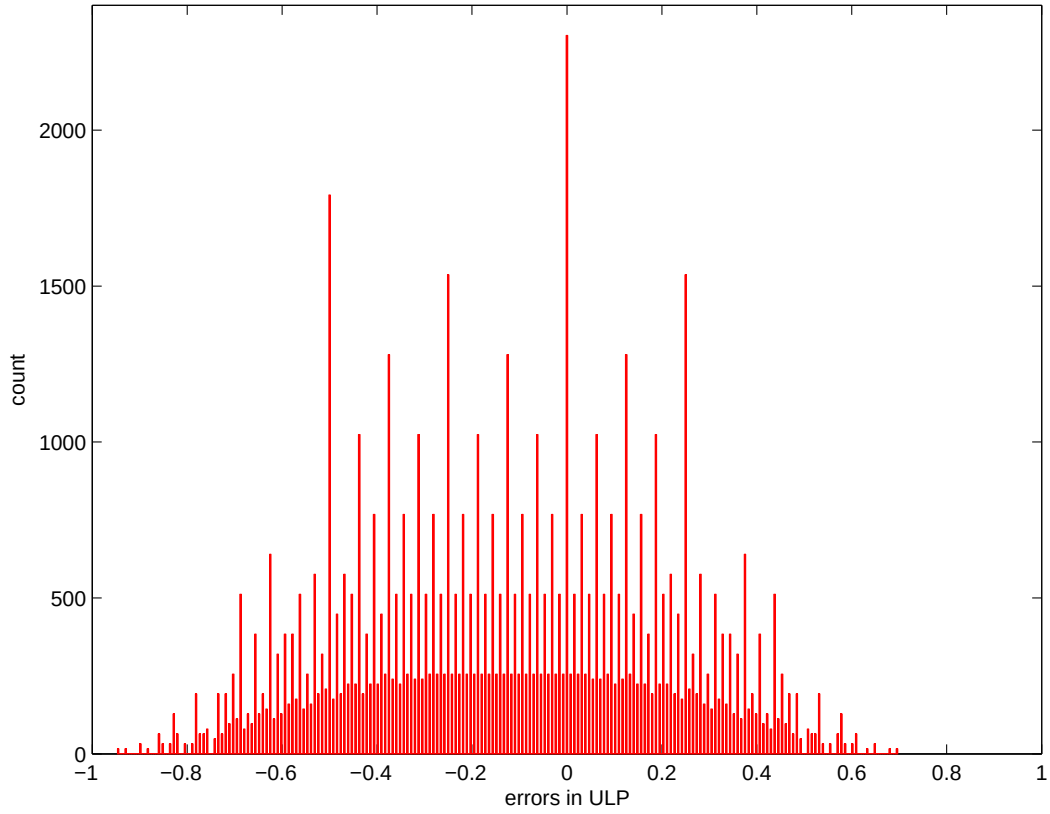

Figure 5.17: Error histogram of 8-bit two's complement truncated multiplication with the variable correction method with k = 1.

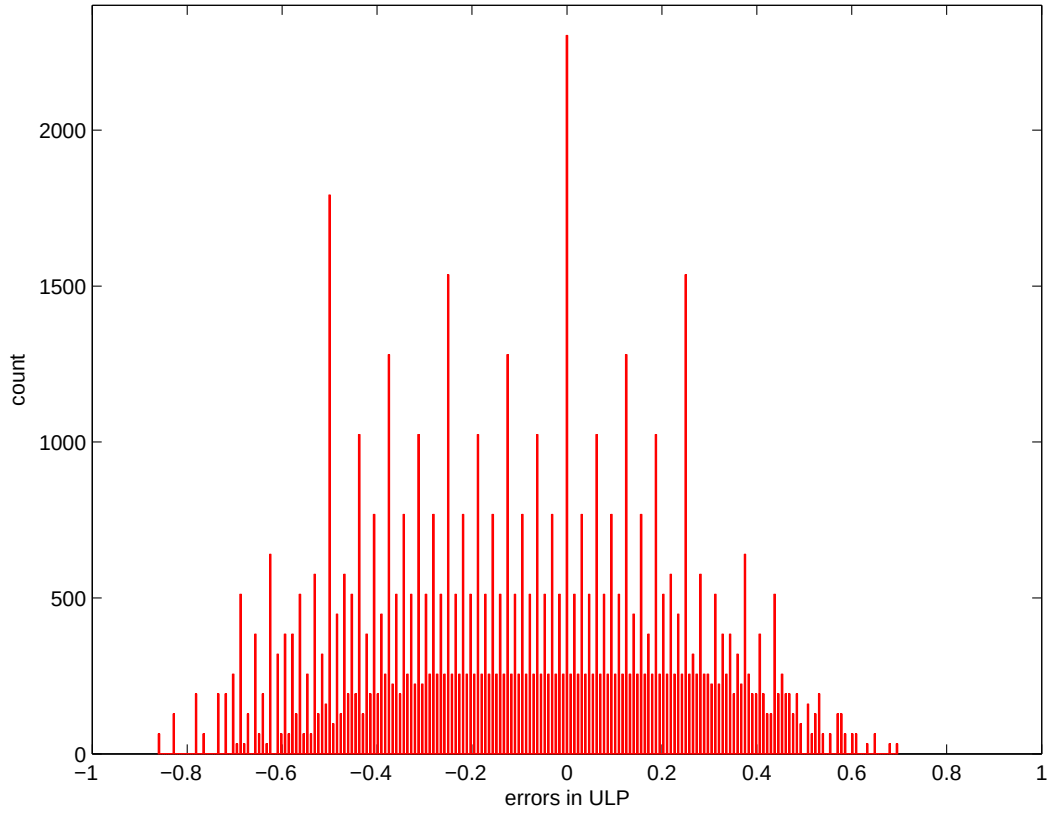

Figure 5.18: Error histogram of 8-bit two's complement truncated multiplication with the new method with k = 1.

5.5 New Truncated Multiplication for Negative Two's Complement Fractions

For multiplication of fixed point signed numbers, the negative two's complement number system is suitable since it is free from an extreme error such as the inherent error (-1 multiplied by -1 yields -1 instead of $+1$) in the conventional two's complement number system. Similar to the unsigned number system and the conventional two's complement number system, new logic can be used to reduce the errors. In the negative two's complement number system, errors of the new correction method in units in the last place is

$$|E_{variable}| \ \leq \ 0.5 + \sum_{i=1}^{\lfloor (n-k-1)/2 \rfloor} (n-k+1-2i) \cdot 2^{-k-2i-1}$$

Maximum errors of the variable correction method in the negative two's complement number system occur if the input patterns meets the condition below.

$$a_{n-k-2} \ ... \ a_0, \ b_{n-k-2} \ ... \ b_0 = \begin{cases} 0 \ x \ \overline{x} \ ... \ x \ \overline{x} \ 1, \ 0 \ x \ \overline{x} \ ... \ x \ \overline{x} \ 1 & \text{if n-k is odd} \\ 0 \ x \ \overline{x} \ ... \ \overline{x} \ x \ 1, \ 0 \ \overline{x} \ x \ ... \ x \ \overline{x} \ 1 & \text{if n-k is even} \end{cases}$$

$$p_{n-1} \ ... \ p_{n-k} = 0 \ ... \ 0$$

Since the negative two's complement number system is viewed as a mirror image of the conventional two's complement number system, the partial product bits have minus sign except the most significant bit and AND and NAND

gates in the partial product are switched. Differently from the unsigned number system and the conventional two's complement number system, maximum errors occur when the partial product bits for error compensation in the n-k-2 column are all 1s and the final product from the n-2 column to the n-k-1 are all ZEROs. This means that error compensation is maximum and result in too many carries to the n most significant columns even though many of the partial product bits, which are not formed, have ZEROs. To prevent this case, error compensation is reduced when a_0 and $b_0 = 1$ and a_{n-k-2} and $b_{n-k-2} = 0$. Subtracting $a_0 b_0 \overline{(a_{n-k-2} + b_{n-k-2})}$ from the error compensation is equal to its complement and -1. -1 can be canceled out with logical 1, one of round constants. Figure 5.19 shows the new correction method in the negative two's complement number system. The special logic is $\overline{a_0 b_0 \overline{(a_{n-k-2} + b_{n-k-2})}}$, which is equal to the complement of $a_0 b_0 \overline{(a_{n-k-2} + b_{n-k-2})}$ in Figure 5.20.

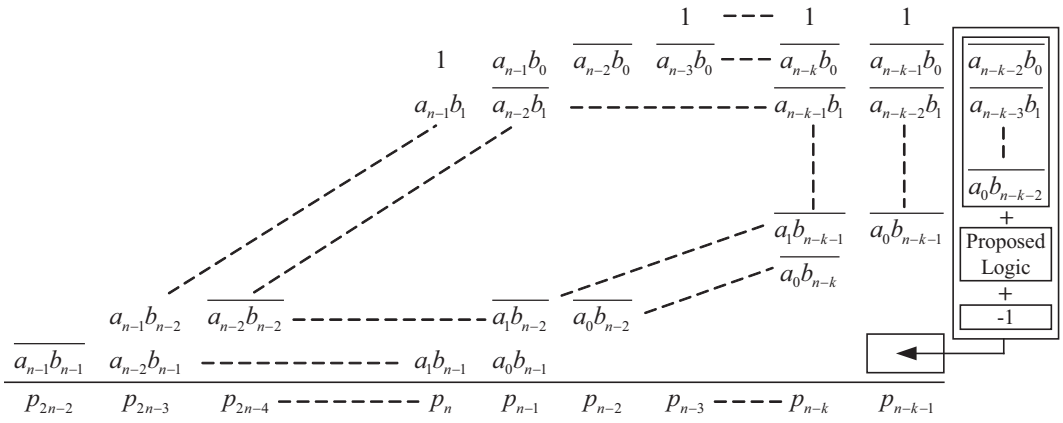

Figure 5.19: The bit product matrix for multiplication in the fractional negative two's complement number system.

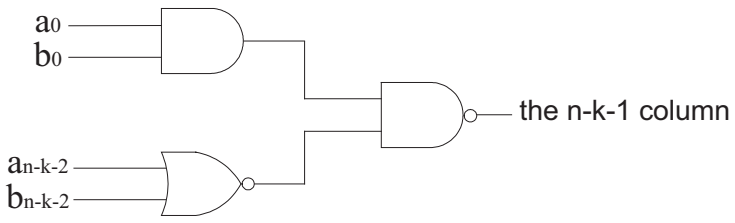

Figure 5.20: Special logic for the new truncated multiplication method in the negative two's complement number system.

5.5.1 Hardware Saving with a Specialized Counter

In contrast to the unsigned number system and the conventional two's complement number system, inputs for the specialized counter are the complement of those in the other number systems. From Table 5.5 and Table 5.6, a specialized (3:2) counter, which has less complexity than a full adder, is proposed. The logical equations for the carry and the sum are

$$Carry = input1 + input2 + \overline{input3}$$

$$Sum = (\overline{input1 \ \oplus \ input2}) \cdot input3$$

When k is greater than 2, round constant in the n-k-1 column cancels -1. In the case of k=1, one logical 1 in the n column should be two logical 1s in the n-1 column and the n-2 column to cancel -1, increasing the hardware for the partial product reduction. To prevent this situation the specialized (3:2) counter is modified. Since the count of $\overline{a_{n-k-2}b_0}$, $\overline{a_0 b_{n-k-2}}$ and $\overline{a_0 b_0 \overline{(a_{n-k-2} + b_{n-k-2})}}$

is from 1 to 3, logical 1 can be borrowed to cancel -1. From Table 5.7, the equations for the carry and the sum of a modified specialized (3:2) counter are

$$Carry = input1 \cdot input2 \cdot input3$$

$$Sum = \overline{(\overline{input1 \ \oplus \ input2}) \cdot input3}$$

Since this (3:2) counter has 0 to 2 as output, it can be extended to a specialized (4:2) counter which yields only one carry.

Table 5.5: Truth table of the partial product bits for error compensation in the negative two's complement number system

a_0	b_0	a_{n-k-2}	b_{n-k-2}	$\overline{a_{n-k-2}b_0}$	$\overline{a_0 b_{n-k-2}}$	$\overline{a_0 b_0 \overline{(a_{n-k-2} + b_{n-k-2})}}$
0	0	0	0	1	1	1
0	0	0	1	1	1	1
0	0	1	0	1	1	1
0	0	1	1	1	1	1
0	1	0	0	1	1	1
0	1	0	1	1	1	1
0	1	1	0	0	1	1
0	1	1	1	0	1	1
1	0	0	0	1	1	1
1	0	0	1	1	0	1
1	0	1	0	1	1	1
1	0	1	1	1	0	1
1	1	0	0	1	1	0
1	1	0	1	1	0	1
1	1	1	0	0	1	1
1	1	1	1	0	0	1

Table 5.6: Truth table of a specialized (3:2) counter in the negative two's complement number system

$\overline{a_{n-k-2}b_0}$	$\overline{a_0 b_{n-k-2}}$	$\overline{a_0 b_0 \overline{(a_{n-k-2} + b_{n-k-2})}}$	Sum	Carry
0	0	0	N/A	N/A
0	0	1	1	0
0	1	0	N/A	N/A
0	1	1	0	1
1	0	0	N/A	N/A
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Table 5.7: Truth table of a modified specialized (3:2) counter in the negative two's complement number system

$\overline{a_{n-k-2}b_0}$	$\overline{a_0 b_{n-k-2}}$	$\overline{a_0 b_0 \overline{(a_{n-k-2} + b_{n-k-2})}}$	Sum	Carry
0	0	0	N/A	N/A
0	0	1	0	0
0	1	0	N/A	N/A
0	1	1	1	0
1	0	0	N/A	N/A
1	0	1	1	0
1	1	0	1	0
1	1	1	0	1

5.5.2 Truncated Array Multipliers with Negative Two's Complement Fractions

Figure 5.21 shows the block diagram of 6-bit negative two's complement truncated array multiplier with constant correction. The NHA cell consists of a half adder and an NAND gate. The SNHA cell is the SHA with an NAND as an input instead of an AND gate.

The block diagram of 6-bit negative two's complement truncated array multiplier with variable correction is shown in Figure 5.22. Half adders along the upper right edge of the constant correction method are replaced by full adders due to the additional partial product bits. The NRFA cell is same to the NFA cell except that the output is only carry out.

Figure 5.23 shows the block diagram of 6-bit negative two's complement truncated array multiplier with the new correction method. As in the unsigned number system, the NRFA cell in the variable correction method is replaced by a specialized (4:2) counter with only carry out for the output in Figure 5.24. The SL cell equals the complement of the special logic in Figure 5.20 to use the same specialized (4:2) counter for the unsigned number system, the conventional two's complement number system and the negative two's complement number system.

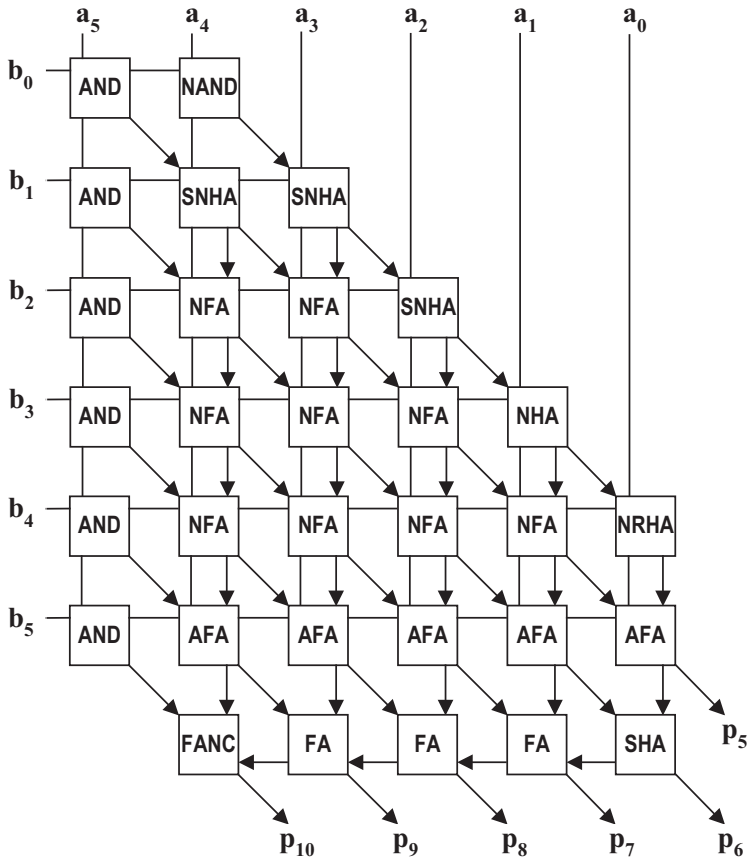

Figure 5.21: 6-bit negative two's complement truncated array multiplier with constant correction.

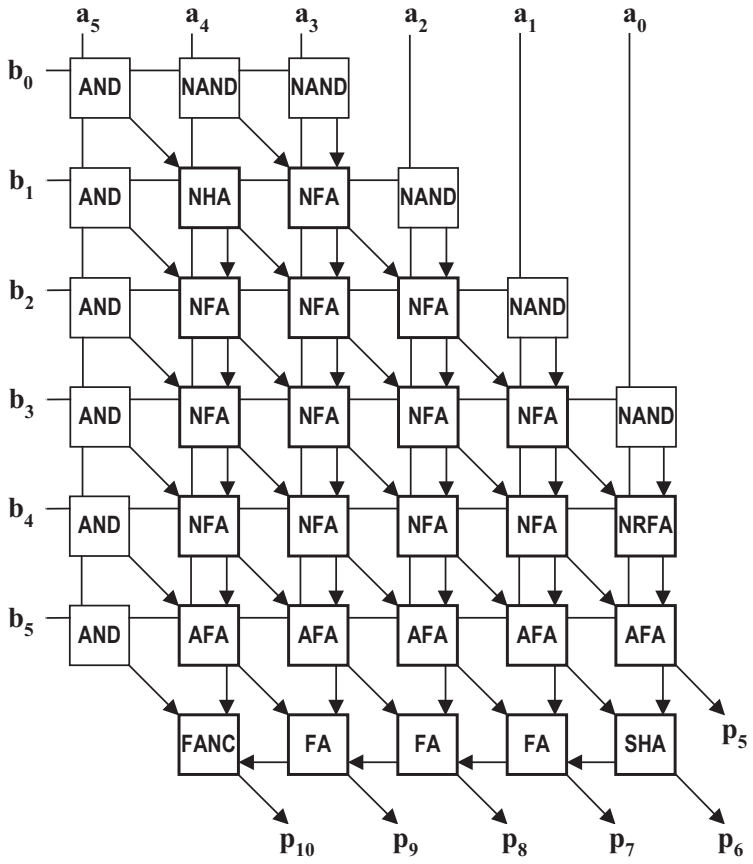

Figure 5.22: 6-bit negative two's complement truncated array multiplier with variable correction.

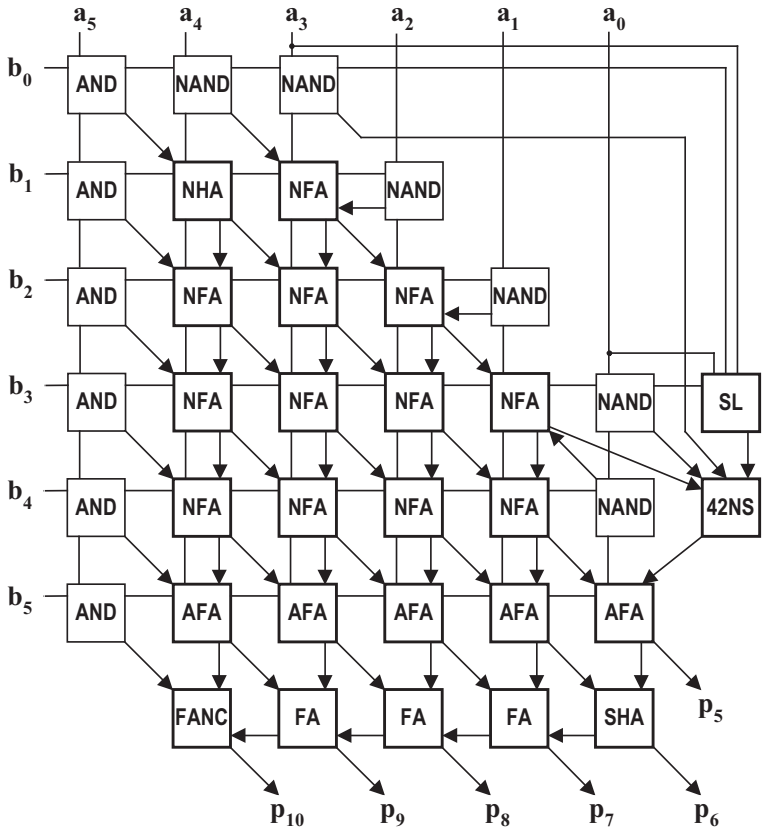

Figure 5.23: 6-bit negative two's complement truncated array multiplier with the new correction method.

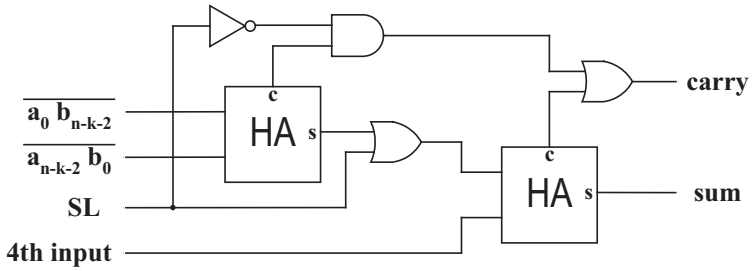

Figure 5.24: Specialized 4:2 counter for negative two's complement array multiplier.

5.5.3 Error Analysis

In the negative two's complement number system, the error of the new correction method in units in the last place is

$$|E_{new}| \ \leq \ 0.5 + \sum_{i=1}^{\lfloor (n-k-2)/2 \rfloor} (n-k-2i) \cdot 2^{-k-2i-1}$$

The error comparison of the variable correction method and the new correction method for all possible input values in the negative two's complement number system is shown in Table 5.8. As in the unsigned number system, maximum error, mean and variance are reduced since the n-1 x n-1 unsigned and the n x n negative two's complement number systems have the same error patterns. When n=9, k=1 and n=16, k=2, the maximum error is less than one unit in the last place. By applying the new correction method, the maximum error is reduced by

$$|D_{maximum}| = \sum_{i=1}^{\lfloor (n-k-3)/2 \rfloor} 2^{-k-2i-1} + 2^{-n+1}$$

Error histograms of 8-bit negative two's complement truncated multiplication with the the constant correction method, the variable correction method and the new method are shown in Figure 5.16, Figure 5.17 and Figure 5.18. Errors by the constant correction method are biased and the absolute value of the negative maximum error is greater than 2. Even though errors by

Table 5.8: Error comparison of variable correction truncation and the new correction truncation method for all possible input values in the negative two's complement number system.

n	k	Max. positive error		Max. negative error		Mean		variance	
		VC	New	VC	New	VC	New	VC	New
7	1	0.609	0.609	-0.859	-0.781	-0.125	-0.094	0.096	0.095
	2	0.516	0.516	-0.641	-0.594	-0.066	-0.051	0.085	0.084
	3	0.484	0.484	-0.547	-0.531	-0.039	-0.031	0.083	0.083
8	1	0.695	0.695	-0.945	-0.859	-0.125	-0.094	0.102	0.101
	2	0.555	0.555	-0.680	-0.641	-0.065	-0.049	0.086	0.086
	3	0.508	0.508	-0.570	-0.547	-0.035	-0.027	0.084	0.084
9	1	0.777	0.777	**-1.027**	**-0.945**	-0.125	-0.094	0.107	0.106
	2	0.598	0.598	-0.723	-0.680	-0.063	-0.048	0.088	0.088
	3	0.527	0.527	-0.590	-0.570	-0.033	-0.025	0.084	0.084
11	1	0.944	0.944	-1.194	-1.111	-0.125	-0.094	0.118	0.117
	2	0.681	0.681	-0.806	-0.764	-0.063	-0.047	0.091	0.090
	3	0.569	0.569	-0.632	-0.611	-0.032	-0.024	0.085	0.085
13	1	1.111	1.111	-1.361	-1.278	-0.125	-0.094	0.128	0.127
	2	0.764	0.764	-0.889	-0.847	-0.063	-0.047	0.093	0.093
	3	0.611	0.611	-0.674	-0.653	-0.031	-0.024	0.085	0.085
16	1	1.361	1.361	-1.611	-1.528	-0.125	-0.094	0.144	0.143
	2	0.888	0.888	**-1.014**	**-0.972**	-0.063	-0.047	0.097	0.097
	3	0.674	0.674	-0.736	-0.715	-0.031	-0.023	0.086	0.086

the variable correction method are less biased and the absolute value of the negative maximum error is less than 1, more negative errors are distributed. By applying the new method, errors are more symmetric than with the other correction methods. The absolute value of the negative maximum error is also decreased and the positive maximum error equals that of the variable correction method. Note that the error distributions for all the methods are the same to those of the conventional two's complement number system.

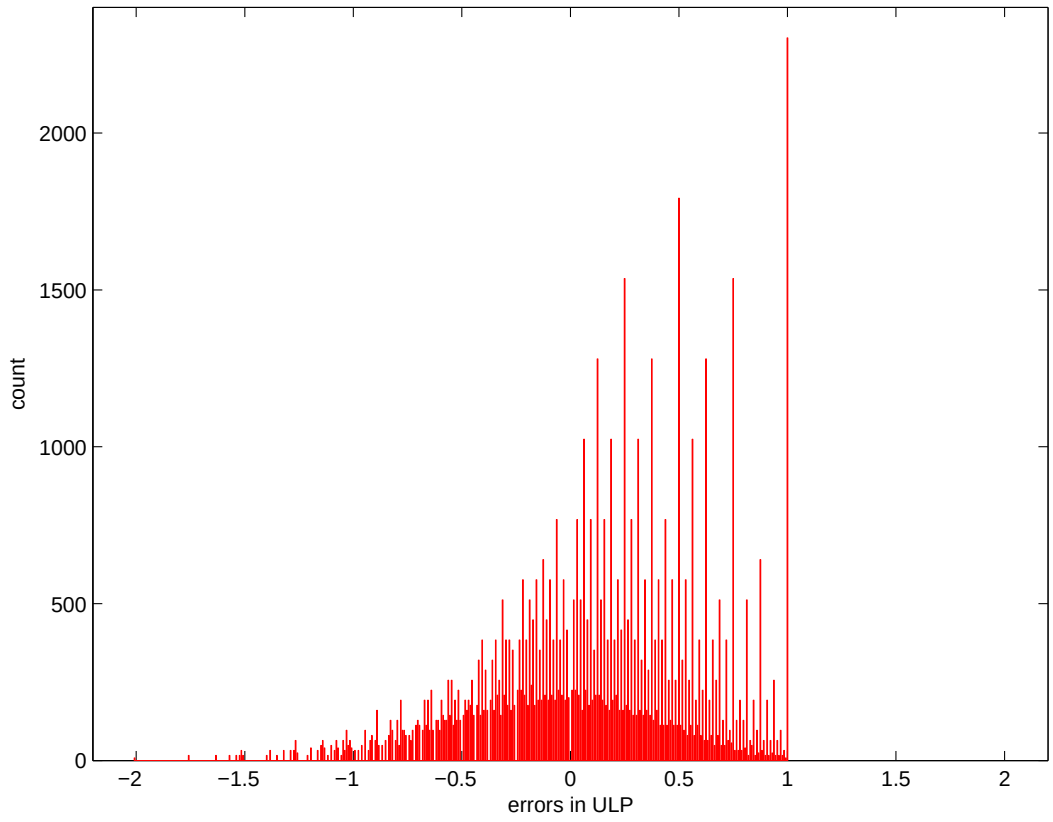

Figure 5.25: Error histogram of 8-bit negative two's complement truncated multiplication with the constant correction method with k = 1.

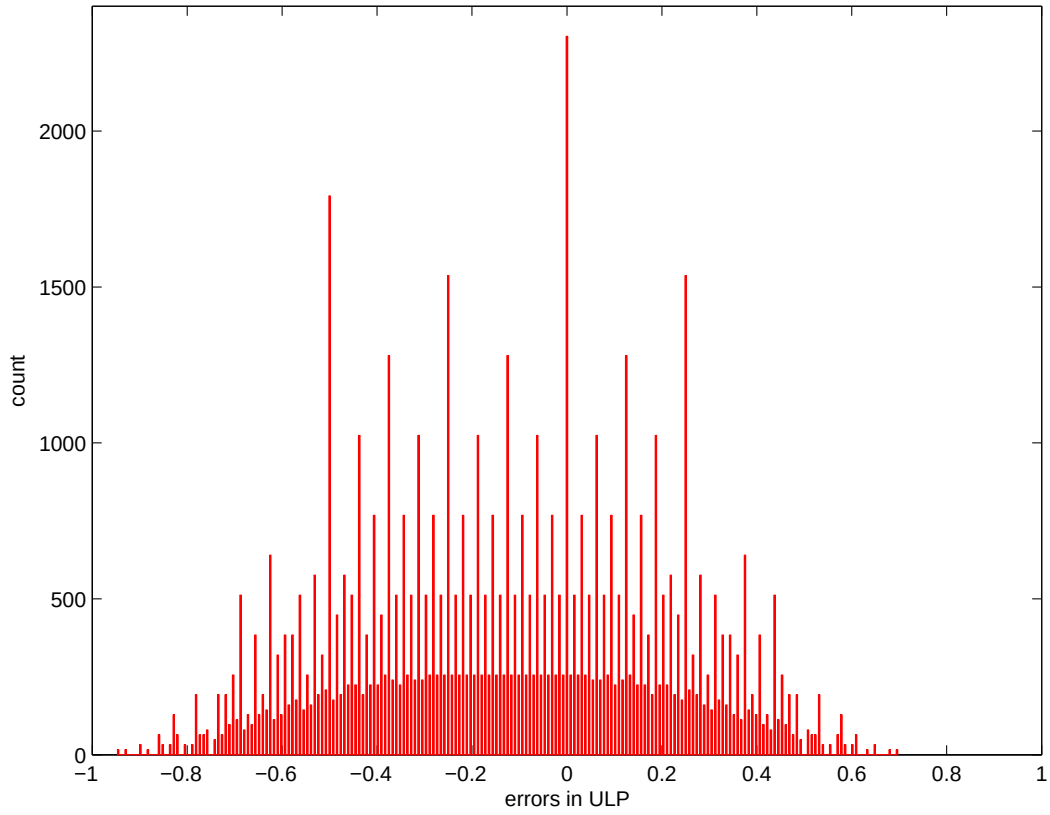

Figure 5.26: Error histogram of 8-bit negative two's complement truncated multiplication with the variable correction method with k = 1.

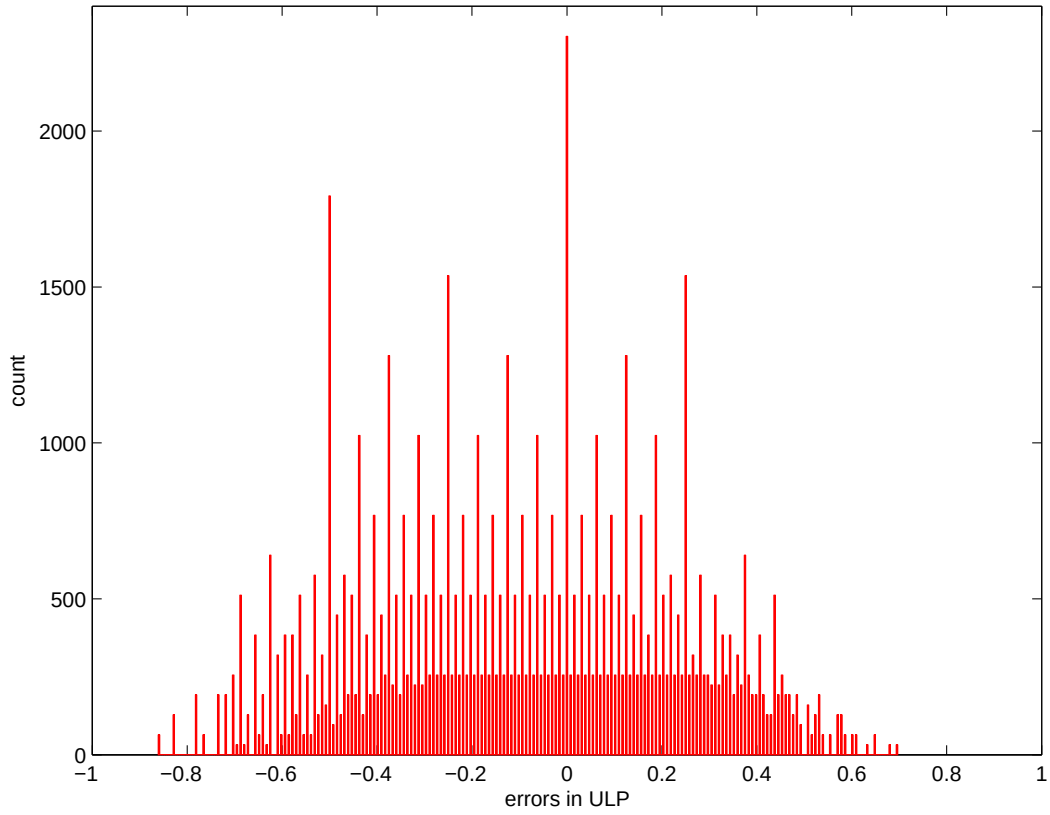

Figure 5.27: Error histogram of 8-bit negative two's complement truncated multiplication with the new method with k = 1.

5.6 Experimental Results

For demonstration of the new method and comparison with the constant correction method and the variable correction method, 8 bit and 16 bit truncated multipliers for the unsigned number system, the conventional two's complement number system and the negative two's complement number system were designed in structural HDL [34, 35], which are automatically generated by perl scripts [36, 37], and synthesized by Synopsys Design Compiler [38] in standard cells for TSMC 0.25 μm technology [39, 40].

5.6.1 Truncated Multipliers with Unsigned Fractions

Table 5.9 shows area, delay and dynamic power of standard array multiplier and truncated array multiplier with constant correction, variable correction and the new method for unsigned numbers. The dynamic power was estimated by capturing the average switching activity of each cell and net for 50,000 random input patterns at the frequency of 10 MHz at 2.5 V. For a maximum absolute error less than 1 ulp in the case of 8 bit x 8 bit multiplication, different values of k are required. As a result, the constant correction method with k = 3 uses 93.95 μW, the variable correction method with k = 2 uses 94.27 μW and the new correction method with k = 1 uses only 80.44 μW.

Table 5.9: Area, delay and dynamic power comparison of standard array multiplier and truncated array multiplier with constant correction, variable correction and the new method for 8 x 8 and 16 x 16 unsigned numbers

n	Multiplier	k	Area (μm^2)		Delay (ns)		Power (μW)	
			Value	Ratio	Value	Ratio	Value	Ratio
8	Standard		28348.19	1	5.69	1	116.28	1
	CCT	1	16901.14	0.60	4.73	0.83	70.19	0.60
		2	19987.43	0.71	4.99	0.88	83.72	0.72
		3	22518.52	0.79	5.24	0.92	93.95	0.81
		4	24510.74	0.86	5.39	0.95	100.29	0.86
	VCT	1	19072.97	0.67	4.93	0.87	79.79	0.69
		2	22420.54	0.79	5.20	0.91	94.27	0.81
		3	24837.32	0.88	5.46	0.96	104.38	0.90
		4	26682.57	0.94	5.72	1.01	111.67	0.96
	New	1	19236.27	0.68	4.93	0.87	80.44	0.69
		2	22698.14	0.80	5.20	0.91	95.35	0.82
		3	25114.93	0.89	5.46	0.96	105.42	0.91
		4	26960.17	0.95	5.72	1.01	112.7	0.97
16	Standard		123583.95	1	12.38	1	541.51	1
	CCT	1	67882.23	0.55	9.45	0.76	294.36	0.54
		2	75149.09	0.61	9.72	0.79	327.34	0.60
		3	81844.4	0.66	9.86	0.80	356.61	0.66
		4	88017.15	0.71	10.01	0.81	382.5	0.71
	VCT	1	72503.63	0.59	9.55	0.77	316.03	0.58
		2	80031.77	0.65	9.82	0.79	350.03	0.65
		3	86629.1	0.70	10.08	0.81	379.64	0.70
		4	92654.88	0.75	10.34	0.84	406.33	0.75
	New	1	72666.92	0.59	9.55	0.77	317.16	0.59
		2	80309.38	0.65	9.82	0.79	351.64	0.65
		3	86906.71	0.70	10.08	0.81	381.22	0.70
		4	92932.49	0.75	10.34	0.84	407.9	0.75

5.6.2 Truncated Multipliers with Two's Complement Fractions

Table 5.10 shows area, delay and dynamic power of standard array multiplier and truncated array multiplier with constant correction, variable correction and the new method for two's complement numbers. For a maximum absolute error less than 1 ulp in the case of 16 bit x 16 bit multiplication, different values of k are required. For a maximum absolute error less than 1 ulp in the case of 16 bit x 16 bit multiplication, different values of k are required. As a result, the constant correction method with k=4 uses 412.39 μW, the variable correction method with k=3 uses 405.95 μW and the new correction method with k=2 uses only 380.76 μW.

5.6.3 Truncated Multipliers with Negative Two's Complement Fractions

Table 5.11 shows area, delay and dynamic power of standard array multiplier and truncated array multiplier with constant correction, variable correction and the new method for negative two's complement numbers. For a maximum absolute error less than 1 ulp in the case of 16 bit x 16 bit multiplication, different values of k are required. For a maximum absolute error less than 1 ulp in the case of 16 bit x 16 bit multiplication, different values of k are required. As a result, the constant correction method with k=4 uses 402.18 μW, the variable correction method with k=3 uses 398.25 μW and the new correction method with k=2 uses only 374.56 μW.

Table 5.10: Area, delay and dynamic power comparison of standard array multiplier and truncated array multiplier with constant correction, variable correction and the new method for 8 x 8 and 16 x 16 two's complement numbers

n	Multiplier	k	Area (μm^2)		Delay (ns)		Power (μW)	
			Value	Ratio	Value	Ratio	Value	Ratio
8	Standard		28315.53	1.00	5.60	1.00	117.58	1.00
	CCT	1	20036.42	0.71	4.79	0.86	81.69	0.69
		2	22616.50	0.80	5.16	0.92	96.73	0.82
		3	24608.71	0.87	5.30	0.95	103.06	0.88
		4	26094.71	0.92	5.56	0.99	108.72	0.92
	VCT	1	21914.32	0.77	5.12	0.91	93.09	0.79
		2	24967.96	0.88	5.38	0.96	104.69	0.89
		3	26862.20	0.95	5.63	1.01	112.31	0.96
		4	28184.89	1.00	5.89	1.05	117.26	1.00
	New	1	22077.62	0.78	5.12	0.91	93.90	0.80
		2	25245.57	0.89	5.38	0.96	105.78	0.90
		3	27139.80	0.96	5.63	1.01	113.36	0.96
		4	28462.50	1.01	5.89	1.05	118.30	1.01
16	Standard		123812.57	1.00	12.28	1.00	543.60	1.00
	CCT	1	75459.37	0.61	9.50	0.77	325.18	0.60
		2	82187.34	0.66	9.77	0.80	354.29	0.65
		3	88376.41	0.71	9.92	0.81	387.49	0.71
		4	94042.93	0.76	10.18	0.83	412.39	0.76
	VCT	1	79786.83	0.64	9.74	0.79	351.31	0.65
		2	87021.02	0.70	10.00	0.81	379.13	0.70
		3	93095.79	0.75	10.25	0.83	405.95	0.75
		4	98599.01	0.80	10.51	0.86	430.10	0.79
	New	1	79950.13	0.65	9.74	0.79	352.61	0.65
		2	87298.63	0.71	10.00	0.81	380.76	0.70
		3	93373.40	0.75	10.25	0.83	407.54	0.75
		4	98876.62	0.80	10.51	0.86	431.62	0.79

Table 5.11: Area, delay and dynamic power comparison of standard array multiplier and truncated multiplication with constant correction, variable correction and the new method for 8 x 8 and 16 x 16 negative two's complement numbers

n	Multiplier	k	Area (μm^2)		Delay (ns)		Power (μW)	
			Value	Ratio	Value	Ratio	Value	Ratio
8	Standard		30536.36	1.00	5.54	1.00	126.58	1.00
	CCT	1	20085.41	0.66	4.66	0.84	83.47	0.66
		2	22681.82	0.74	4.80	0.87	97.42	0.77
		3	24788.34	0.81	5.05	0.91	101.37	0.80
		4	26339.65	0.86	5.31	0.96	107.39	0.85
	VCT	1	21995.97	0.72	4.84	0.87	89.14	0.70
		2	25131.26	0.82	5.10	0.92	104.33	0.82
		3	27090.81	0.89	5.35	0.97	113.29	0.90
		4	28462.5	0.93	5.61	1.01	119.61	0.94
	New	1	22159.27	0.73	4.84	0.87	90.35	0.71
		2	25408.86	0.83	5.10	0.92	106.13	0.84
		3	27368.42	0.90	5.35	0.97	114.88	0.91
		4	28740.10	0.94	5.61	1.01	121.06	0.96
16	Standard		130213.88	1.00	11.94	1.00	545.06	1.00
	CCT	1	76553.48	0.59	9.39	0.79	319.74	0.59
		2	83444.73	0.64	9.40	0.79	353.42	0.65
		3	89780.77	0.69	9.41	0.79	386.67	0.71
		4	95675.91	0.73	9.73	0.81	402.18	0.74
	VCT	1	80978.90	0.62	9.17	0.77	335.40	0.62
		2	88425.38	0.68	9.43	0.79	370.98	0.68
		3	94696.10	0.73	9.69	0.81	398.25	0.73
		4	100378.95	0.77	9.95	0.83	422.76	0.78
	New	1	81142.20	0.62	9.17	0.77	338.34	0.62
		2	88702.98	0.68	9.43	0.79	374.56	0.69
		3	94973.71	0.73	9.69	0.81	401.55	0.74
		4	100656.55	0.77	9.95	0.83	425.79	0.78

5.7 Summary

In many signal processing applications, multiplication with approximate rounding can significantly reduce the power dissipation and area by truncating the least significant part of the bit product matrix and compensating for the errors. In this chapter, a new approximate rounding method has been proposed and compared to the previous methods showing reduced maximum errors, reduced mean error and reduced variances with very slightly increased power, delay and area. This method is implemented very efficiently by the use of a specialized counter.

Chapter 6

Truncated Division

In the design of many systems, it is desirable to reduce the execution time of division since it is the slowest of the basic arithmetic operations. In digit-recurrence division, a rounded result is created by an additional iteration and carry propagate addition, increasing the execution time. A solution is truncated division which presets the round value during division without increase of the execution time. This chapter extends truncated division with the constant round value [6] to the floating point number system, two's complement number system and the negative two's complement number system.

6.1 On the Fly Conversion

In digit-recurrence division, the quotient is generated by sequences of shifts and adds. When each iteration is performed in series, each quotient digit is also produced in series. If the quotient digit is signed, the digit needs to be converted into a binary number format by subtracting the negative digits from the positive digits. In Figure 6.1, the positive digits and the negative digits are separated before the negative digits are converted into two's complement

number format. The converted digits are added to the positive digits. This means a carry propagate addition is required due to conversion of the negative digits and addition of the two numbers.

$$
\begin{aligned}
Q &= 0.1\bar{1}11\bar{1}1 = 27/64 \\
n &= 0.010010 \quad \text{formation of n} \\
p &= 0.101101 \quad \text{formation of p} \\
p - n &= 0.101101 - 0.010010 \\
&= 0.101101 + 1.101110 \quad \text{addition} \\
&= 0.011011 = 27/64
\end{aligned}
$$

Figure 6.1: Conventional conversion of signed digits into a binary number [13].

Since no restoration is necessary in non-restoring division, it is faster than restoring division, but, the remainder after n iterations can be negative. Therefore, a correction step, in which the remainder after n iterations is added to the divisor and the quotient is decreased by 1, may be required to insure a positive remainder. Furthermore, the quotient in signed digit form needs to be converted to standard binary form.

By conversion on the fly as the digits of the quotient are obtained [10], the correction step is performed without carry-propagate addition and the signed quotient digits are converted to standard binary form in a digit-serial manner as the quotient digits are produced. In fractional binary non-restoring division, the signed quotient is $q_{n-1}q_{n-2}...q_0$ with $q_k \in \{-1, 1\}$. When all signed

quotient digits with -1 are replaced by 0, the MSB is complemented and then all quotient digits are left shifted by 1 bit, inserting 1 into the LSB, to get the two's complement quotient $\overline{q'_{n-1}}q'_{n-2}...q'_0 1$ with $q'_k \in \{0, 1\}$. Note that $\overline{q'_{n-1}}$ is the sign bit and the LSB is always 1. In the correction step, the LSB is 0 or 1 without carry propagate addition since the quotient is only decreased by 1 when the remainder after n iterations is negative. Figure 6.2 shows an example of on the fly conversion. Note that all procedures for conversion is performed when each signed digit is obtained without increased execution time for addition.

$$
\begin{aligned}
Q = .1\bar{1}11\bar{1}1 & \\
.101101 \quad & \text{Replace -1 by 0} \\
.001101 \quad & \text{Complement MSB} \\
.01101 \quad & \text{Left-shift by 1} \\
= .011011 \quad & \text{Insert 1 into LSB}
\end{aligned}
$$

Figure 6.2: On the fly conversion of signed digits into a binary number [10].

The on the fly conversion can be extended to rounding. By an additional iteration, a signed quotient $q_{n-1}q_{n-2}...q_0q_{-1}$ with $q_k \in \{$-1, 1$\}$ is produced. This signed quotient is converted to a two's complement quotient $\overline{q'_{n-1}}q'_{n-2}...\ q'_0p_{-1}1$. Note that q'_{-1} is the LSB and 1 is a round bit. For the correction step, the quotient is decreased by 1 when the remainder after n+1

iterations is negative as shown above. For true rounding, the quotient is increased by 1 when the remainder after n+1 iterations is positive. Alternatively, the quotient is always increased by 1 for true rounding. Note that both ways produce the same rounded n bit quotient and both require carry propagate addition.

6.2 Truncated Division in Various Number Systems

In truncated division with constant correction for the unsigned number system, 1/3 ulp as a constant round value is preset into the dividend to get an approximately rounded result when the divisor is normalized in range of [1/2, 1) in [6]. For the floating point number system, the two's complement number system and the negative two's complement number system, the constant round values are examined.

6.2.1 Constant Truncated Division for Floating Point Fractions

In the division of floating point numbers, the sign, the significand and the exponent of the quotient are calculated separately. The two signs are ex-ORed and the two exponents are subtracted and added to the bias [41]. Here the focus is on the division of the significands. Since for numbers conforming to IEEE Std. 754 [42], the dividend and the divisor are normalized to the range of [1, 2), the quotient will be in the range (1/2, 2). When $N \geq D$, the quotient is in the range [1, 2) which is normalized. When $N < D$, the quotient

is in the range (1/2, 1) which requires that the quotient be left shifted by 1 bit position (and also requires that the exponent be decremented by 1) to get the quotient in the range (1, 2). In this case, an extra iteration for the guard bit is required. An alternative way is to left-shift the dividend before division when N < D with no extra iteration for the guard bit, but this requires a pre-comparison of the dividend and the divisor. Differently from the unsigned number system, the divisor is in the range of [1, 2). Therefore the preset value to the dividend is 2/3 ulp instead of 1/3 ulp. In n-bit radix-2 floating point fractional non-restoring division with the constant method,

$$P_{k+1} = 2P_k + 2^{-n} - q_{n-k-1} \cdot D \quad \textit{if } k \textit{ is even for } k = 0, 1, ..., n-1$$
$$P_{k+1} = 2P_k - q_{n-k-1} \cdot D \quad \textit{if } k \textit{ is odd for } k = 0, 1, ..., n-1$$

where P_k is the partial remainder after selecting the kth quotient digit, $P_0 = \mathrm{N}/2$ when $\mathrm{N} \geq \mathrm{D}$ and $P_0 = \mathrm{N}$ when $\mathrm{N} < \mathrm{D}$. The quotient digits can be -1 or $+1$ and the digit for the kth iteration is selected by

$$\textit{If } P_k \geq 0, \ q_{n-k-1} = +1$$
$$\textit{If } P_k < 0, \ q_{n-k-1} = -1$$

After n iterations and on the fly conversion, the LSB is converted to 0 when the final partial remainder, P_n, is negative. Note that the sign bit, $\overline{q'_{n-1}}$, is always 0.

$$\textit{If } P_n < 0, \ Q = \overline{q'_{n-1}} q'_{n-2} ... q'_0 0$$

$$Otherwise,\ Q = \overline{q'_{n-1}}q'_{n-2}...q'_0 1$$

6.2.2 Constant Truncated Division for Two's Complement Fractions

As for the unsigned number system, 1/3 ulp as a constant round value is preset into the dividend to get an approximately rounded result. Even though the normalized divisor is in range of [1/2, 1) or [-1, -1/2], the fraction bits of the dividend are positive regardless of the sign of the dividend and the divisor. Therefore, 1/3 ulp is preset into the dividend in both normalized divisor range of [1/2, 1) or [-1, -1/2]. In n-bit radix-2 two's complement fractional non-restoring division with the constant method,

$$P_{k+1} = 2P_k - q_{n-k-1} \cdot D \quad if\ k\ is\ even\ for\ k = 0, 1, ..., n-2$$
$$P_{k+1} = 2P_k + 2^{-n+1} - q_{n-k-1} \cdot D \quad if\ k\ is\ odd\ for\ k = 0, 1, ..., n-2$$

where P_k is the partial remainder after selecting the kth quotient digit, $P_0 = N$, and $|N| < |D|$. The quotient digits can be -1 or $+1$ and the digit for the kth iteration is selected by

$$If\ sign\ of\ P_k\ =\ sign\ of\ D,\ q_{n-k-1} = +\ 1$$
$$If\ sign\ of\ P_k \neq\ sign\ of\ D,\ q_{n-k-1} = -\ 1$$

After n-1 iterations and on the fly conversion, the correction step is performed by increasing or decreasing the quotient by 1 when the final partial

remainder, P_{n-1}, is negative.

$$\textit{If } P_{n-1} < 0 \textit{ and } D > 0,\ Q = \overline{q'_{n-1}} q'_{n-2} ... q'_1 0$$

$$\textit{If } P_{n-1} < 0 \textit{ and } D < 0,\ Q = \overline{q'_{n-1}} q'_{n-2} ... q'_1 1 + 1$$
$$\textit{Otherwise},\ Q = \overline{q'_{n-1}} q'_{n-2} ... q'_1 1$$

When the divisor is negative, the correction step may need carry propagate addition since the quotient needs to be increased by 1, increasing execution time, but it can be avoided by swapping the signs of the divisor and the dividend when the divisor is negative. After swapping the signs of the divisor and the dividend to get positive divisor,

$$\textit{If } P_{n-1} < 0,\ Q = \overline{q'_{n-1}} q'_{n-2} ... q'_1 0$$
$$\textit{Otherwise},\ Q = \overline{q'_{n-1}} q'_{n-2} ... q'_1 1$$

6.2.3 Constant Truncated Division for Negative Two's Complement Fractions

In the two's complement fractional number system, -1 multiplied by -1 yields -1 instead of $+1$. The negative two's complement fractional number system, which expresses values for $1 \geq \mathrm{V} > -1$, solves this problem [5]. As for the unsigned number system and the two's complement number system, 1/3 ulp as a constant round value is preset into the dividend to get an approximately rounded result. Any preset value into the dividend has a negative

sign, and the fraction bits of the dividend have also negative signs regardless of the sign of the dividend and the divisor. Therefore, a 1/3 ulp is preset into the dividend and in both normalized divisor range of [1/2, 1] or (-1, -1/2]. In n-bit radix-2 negative two's complement fractional non-restoring division with the constant method,

$$P_{k+1} = 2P_k - q_{n-k-1} \cdot D \quad \textit{if } k \textit{ is even for } k = 0, 1, ..., n-2$$
$$P_{k+1} = 2P_k - 2^{-n+1} - q_{n-k-1} \cdot D \quad \textit{if } k \textit{ is odd for } k = 0, 1, ..., n-2$$

where P_k is the partial remainder after selecting the kth quotient digit, $P_0 = N$, and $|N| < |D|$. The quotient digits can be -1 or $+1$ and the digit for the kth iteration is selected by

$$\textit{If sign of } P_k \neq \textit{ sign of } D,\ q_{n-k-1} = +1$$
$$\textit{If sign of } P_k = \textit{ sign of } D,\ q_{n-k-1} = -1$$

After n-1 iterations and on the fly conversion, the correction step is performed by increasing or decreasing the quotient by 1 when the final partial remainder, P_{n-1}, is positive.

$$\textit{If } P_{n-1} > 0 \textit{ and } D > 0,\ Q = \overline{q'_{n-1}} q'_{n-2} ... q'_1 0$$

$$\textit{If } P_{n-1} > 0 \textit{ and } D < 0,\ Q = \overline{q'_{n-1}} q'_{n-2} ... q'_1 1 + 1$$
$$\textit{Otherwise},\ Q = \overline{q'_{n-1}} q'_{n-2} ... q'_1 1$$

As with the two's complement number system, a carry propagate addition may be required when the divisor is negative, but it can be avoided by swapping the signs of the divisor and the dividend when the divisor is negative. Note that negative numbers in the negative two's complement number system have 0 for the sign bit. After swapping the signs of the divisor and the dividend to get positive divisor,

$$If\ P_{n-1} < 0,\ Q = \overline{q'_{n-1}} q'_{n-2} ... q'_1 0$$

$$Otherwise,\ Q = \overline{q'_{n-1}} q'_{n-2} ... q'_1 1$$

6.3 Summary

In truncated division with constant correction for the unsigned number system, each bit of a round constant value, 1/3 ulp, is preset to the dividend during each iteration to get an approximately rounded result in the range of (-2/3, 2/3). For the two's complement number system and the negative two's complement number system, the same round constant round value of 1/3 ulp is preset to the dividend. For the floating point number system, a different round constant round value of 2/3 ulp is preset to the dividend.

Chapter 7

Truncated Division with Concurrent Rounding

To get a rounded result in digit-recurrence division, an additional iteration and carry propagate addition are required, increasing the execution time. This chapter suggests a new method for truncated division which yields the same errors as those of true rounding with a small hardware increase without an additional iteration and carry propagate addition. The new truncation method is applied to the unsigned number system, the floating point number system, the two's complement number system and the negative two's complement number system. Errors of the previous methods and the new truncation method are compared via exhaustive simulation.

7.1 Overview

In digit-recurrence division, one digit of the quotient is generated by each iteration of shifting and addition [12]. For a rounded quotient result, the extra iteration and addition for true rounding are required resulting in an increased execution time. To mitigate the increase of the execution time, a constant round value, 1/3 ulp, may be preset into the dividend in order to

effect an increase in value of the final quotient so that when the quotient result is truncated (without the extra iteration and addition for true rounding) an approximately rounded result is obtained [6]. When the divisor is normalized before division, the errors are bounded between -2/3 ulp and 2/3 ulp.

For more accurate results than the constant truncation method, this paper identifies variable truncation, a new method for digit-recurrence division. It achieves errors bounded between -1/2 ulp and 1/2 ulp (the same as those obtained with true rounding). This method does not require the extra iteration and addition for true rounding nor does it need a normalized divisor. The new method is applied to non-restoring division for unsigned numbers, two's complement numbers, negative two's complement numbers and floating point numbers. Comparison of the new truncation method to the previous methods in terms of error and the numbers of iteration and addition is given.

7.2 New Truncated Division

Even though the constant truncated division reduces the execution time, the divisor needs to be normalized and the errors are larger than those of true rounded division. To overcome these drawbacks, a new truncated division method can be used. For true rounding, 1 is added to the extra quotient bit (round bit) or 1/2 is added to the LSB of the quotient after the additional iteration. It is equivalent to preset half of the divisor into the LSB of the dividend since a half of divisor ulp in the dividend divided by the divisor is always

1/2 ulp of the quotient, which is combined with the partial remainder. This variable round value is computed during division and the effect on the quotient is error that is bounded between -1/2 and 1/2 (the same as true rounding) without an additional iteration, carry propagate addition and normalization of the divisor.

7.2.1 New Truncated Division for the Unsigned Fractions

In n-bit radix-2 unsigned fractional non-restoring division with the new method,

$$P_{k+1} = 2P_k + d_{n-k}2^{-n} - q_{n-k-1} \cdot D \quad for\ k = 0, 1, ..., n-1$$

$$D = \sum_{k=0}^{n-1} d_{n-k-1}2^{-k-1},\ d_n = 0$$

where P_k is the partial remainder after selecting the kth quotient digit and $P_0 = N < D$. The quotient digits can be -1 or $+1$ and the digit for the kth iteration is selected by

$$If\ P_k \geq 0,\ q_{n-k-1} = +\ 1$$
$$If\ P_k < 0,\ q_{n-k-1} = -\ 1$$

After n iterations and on the fly conversion, the LSB is converted to 0 when the final partial remainder, P_n, is negative. Note that in unsigned division, the sign bit, $\overline{q'_{n-1}}$, is always 0.

$$If\ P_n < 0,\ Q = \overline{q'_{n-1}}q'_{n-2}...q'_0 0$$

$$Otherwise,\ Q = \overline{q'_{n-1}} q'_{n-2} ... q'_0 1$$

Figure 7.1 shows a hardware diagram of the sequential non-restoring division. At the first cycle, the sign of the quotient is given by the Exclusive-OR of the signs of the dividend and the divisor. Except for the first cycle, the complemented sign of the partial remainder (c_{out} from the n-bit adder) and the sign of the divisor are XORed to get the quotient bits, which is equivalent to the XNOR of the signs of the partial remainder and the divisor. After n iterations, the LSB is 0 or 1 after the correction step.

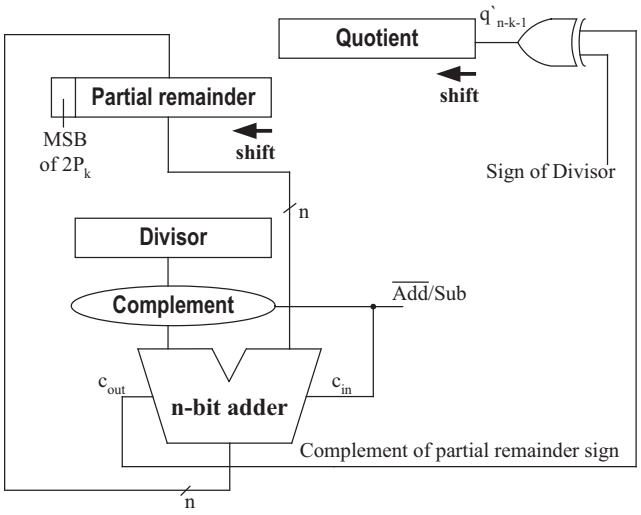

Figure 7.1: Shift/add sequential non-restoring divider [10].

Figure 7.2 shows contents of the shifted partial remainder in k_{th} iteration of Figure 7.1 in standard division, constant truncated division and new truncated division. Since the only difference between standard division, constant truncated division and new truncated division is the contents of the

shifted partial remainder, it is straightforward to compare their overall hardware complexity by comparing the contents of their shifted partial remainder registers. In contrast to standard division, a value is preset to the partial remainder for each iteration in both constant truncated division and new truncated division. Since the preset value is just one bit for each iteration, the hardware increase is quite small, but the preset value prevents an additional iteration and carry propagate addition for true rounding.

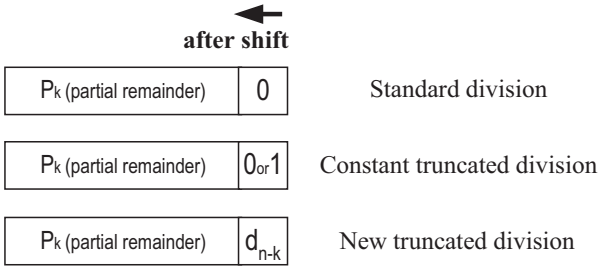

Figure 7.2: Contents of shifted partial remainder in k_{th} iteration of standard division, constant truncated division and new truncated division.

7.2.2 New Truncated Division for Floating Point Fractions

In the division of floating point numbers, the sign, the significand and the exponent of the quotient are calculated separately. The two signs are ex-ORed and the two exponents are subtracted and added to the bias [41]. Here the focus is on the division of the significands. Since for numbers conforming to IEEE Std. 754 [42], the dividend and the divisor are normalized to the range of [1, 2), the quotient will be in the range (1/2, 2). When N $\geq$ D, the quotient is in the range [1, 2) which is normalized. When N $<$ D, the quotient

is in the range (1/2, 1) which requires that the quotient be left shifted by 1 bit position (and also requires that the exponent be decremented by 1) to get the quotient in the range (1, 2). In this case, an extra iteration for the guard bit is required. An alternative way is to left-shift the dividend before division when N < D with no extra iteration for the guard bit, but this requires a pre-comparison of the dividend and the divisor.

In n-bit radix-2 floating point fractional non-restoring division with the new truncation method,

$$P_{k+1} = 2P_k + d_{n-k}2^{-n} - q_{n-k-1} \cdot D \quad for\ k = 0, 1, ..., n-1$$

$$D = d_{n-1} + \sum_{k=0}^{n-2} d_{n-k-2}2^{-k-1},\ d_n = 0$$

where P_k is the partial remainder after selecting the kth quotient digit, $P_0 = \mathrm{N}/2$ when $\mathrm{N} \geq \mathrm{D}$ and $P_0 = \mathrm{N}$ when $\mathrm{N} < \mathrm{D}$. The quotient digits can be -1 or $+1$ and the digit for the kth iteration is selected by

$$If\ P_k \geq 0,\ q_{n-k-1} = +\ 1$$
$$If\ P_k < 0,\ q_{n-k-1} = -\ 1$$

After n iterations and on the fly conversion, the LSB is converted to 0 when the final partial remainder, P_n, is negative. Note that the sign bit, $\overline{q'_{n-1}}$, is always 0.

$$If\ P_n < 0,\ Q = \overline{q'_{n-1}}q'_{n-2}...q'_0 0$$

$$Otherwise,\ Q = \overline{q'_{n-1}}q'_{n-2}...q'_0 1$$

7.2.3 New Truncated Division for Two's Complement Fractions

In n-bit radix-2 two's complement fractional non-restoring division with the new truncation method,

$$P_{k+1} = 2P_k + d_{n-k-1}2^{-n+1} - q_{n-k-1} \cdot D$$

$$if\ D \geq 0\ for\ k = 0, 1, ..., n-2$$

$$P_{k+1} = 2P_k + \overline{d_{n-k-1}}2^{-n+1} - q_{n-k-1} \cdot D$$

$$if\ D < 0\ for\ k = 0, 1, ..., n-2$$

$$D = -d_{n-1} + \sum_{k=0}^{n-2} d_{n-k-2}2^{-k-1}$$

where P_k is the partial remainder after selecting the kth quotient digit, $P_0 = N$, and $|N| < |D|$. When the divisor is negative, the variable round value is a half of -D since all the dividend and the quotient bits except the MSBs are always positive. The quotient digits can be -1 or $+1$ and the digit for the kth iteration is selected by

$$If\ sign\ of\ P_k = sign\ of\ D,\ q_{n-k-1} = +1$$

$$If\ sign\ of\ P_k \neq sign\ of\ D,\ q_{n-k-1} = -1$$

After n-1 iterations and on the fly conversion, the correction step is performed by increasing or decreasing the quotient by 1 when the final partial

remainder, P_{n-1}, is negative.

$$If\ P_{n-1} < 0\ and\ D > 0,\ Q = \overline{q'_{n-1}}q'_{n-2}...q'_1 0$$

$$If\ P_{n-1} < 0\ and\ D < 0,\ Q = \overline{q'_{n-1}}q'_{n-2}...q'_1 1 + 1$$
$$Otherwise,\ Q = \overline{q'_{n-1}}q'_{n-2}...q'_1 1$$

Carry propagate addition may be required when the divisor is negative, but it can be avoided by swapping the signs of the divisor and the dividend when the divisor is negative.

7.2.4 New Truncated Division for Negative Two's Complement Fractions

In the two's complement fractional number system, -1 multiplied by -1 yields -1 instead of $+1$. The negative two's complement fractional number system, which expresses values for $1 \geq \mathrm{V} > -1$, solves this problem [5]. In n-bit radix-2 negative two's complement fractional non-restoring division with the new truncation method,

$$P_{k+1} = 2P_k - \overline{d_{n-k-1}}2^{-n+1} - q_{n-k-1} \cdot D$$
$$if\ D > 0\ for\ k = 0, 1, ..., n-2$$
$$P_{k+1} = 2P_k - d_{n-k-1}2^{-n+1} - q_{n-k-1} \cdot D$$
$$if\ D < 0\ for\ k = 0, 1, ..., n-2$$

$$D = d_{n-1} - \sum_{k=0}^{n-2} d_{n-k-2} 2^{-k-1}$$

where P_k is the partial remainder after selecting the kth quotient digit, $P_0 = N$, and $|N| < |D|$. When the divisor is positive, the variable round value is a half of -D since all the dividend and the quotient bits except the MSBs are always negative. The quotient digits can be -1 or $+1$ and the digit for the kth iteration is selected by

$$\textit{If sign of } P_k \neq \textit{sign of } D,\ q_{n-k-1} = +\ 1$$
$$\textit{If sign of } P_k = \textit{sign of } D,\ q_{n-k-1} = -\ 1$$

After n-1 iterations and on the fly conversion, the correction step is performed by increasing or decreasing the quotient by 1 when the final partial remainder, P_{n-1}, is positive.

$$\textit{If } P_{n-1} > 0 \textit{ and } D > 0,\ Q = \overline{q'_{n-1}} q'_{n-2} \ldots q'_1 0$$

$$\textit{If } P_{n-1} > 0 \textit{ and } D < 0,\ Q = \overline{q'_{n-1}} q'_{n-2} \ldots q'_1 1 + 1$$
$$\textit{Otherwise},\ Q = \overline{q'_{n-1}} q'_{n-2} \ldots q'_1 1$$

As with the two's complement number system, a carry propagate addition may be required when the divisor is negative, but it can be avoided by swapping the signs of the divisor and the dividend when the divisor is negative.

7.3 Experimental Results

For the unsigned number system, the floating point number system, the two's complement number system and the negative two's complement number system, binary non-restoring divisions with true rounding, constant truncation and the new truncation method were simulated in C with all possible inputs of the dividend and the divisor. All error factors are in units in the last place. Table 7.1 shows that true rounding and the new truncation method have the same error patterns which is bounded between -1/2 and 1/2. In constant truncation, errors are bounded between -2/3 and 2/3. The two's complement number system and the negative two's complement number system have the same errors. In the floating point number system, the maximum positive error and the maximum negative error are slightly asymmetric. The maximum positive error occurs when the divisor is one unit less than 2 and the dividend is 1. For the maximum negative error to have the same absolute value as the maximum positive error would require the divisor to be one unit less than 2 and the dividend to be one unit less than 1, which is not possible since the dividend is required to be in the range of [1, 2) in the floating point number system.

Table 7.2 compares execution time of n-bit binary non-restoring division with true rounding, constant truncation and new truncation for unsigned numbers, floating point numbers, two's complement numbers and negative two's complement numbers. True rounding requires an additional iteration for the round bit and carry propagate addition in the correction step. In the

Table 7.1: Error comparison of binary non-restoring division with true rounding, constant truncation and new truncation for unsigned numbers, floating point numbers, two's complement numbers and negative two's complement numbers

Bits	Rounding method	Error factors	Number System			
			Unsigned	FLP	2's comp.	Neg. 2's comp.
6	True rounding	Max. pos. error	.4921	.4920	.4839	.4839
		Max. neg. error	-.4921	-.4915	-.4839	-.4839
		Mean	.0000	.0195	.0000	.0000
		Variance	.0800	.0800	.0766	.0766
	Constant truncation	Max. pos. error	.6364	.6364	.6452	.6452
		Max. neg. error	-.6508	-.6508	-.6452	-.6452
		Mean	-.0521	-.0264	.0000	.0000
		Variance	.0872	.0904	.0858	.0858
	New truncation	Max. pos. error	.4921	.4920	.4839	.4839
		Max. neg. error	-.4921	-.4915	-.4839	-.4839
		Mean	.0000	.0195	.0000	.0000
		Variance	.0800	.0800	.0766	.0766
8	True rounding	Max. pos. error	.4980	.4980	.4961	.4961
		Max. neg. error	-.4980	-.4980	-.4961	-.4961
		Mean	.0000	.0067	.0000	.0000
		Variance	.0825	.0826	.0816	.0816
	Constant truncation	Max. pos. error	.6589	.6589	.6614	.6614
		Max. neg. error	-.6627	-.6627	-.6614	-.6614
		Mean	-.0547	-.0354	.0000	.0000
		Variance	.0902	.0920	.0921	.0921
	New truncation	Max. pos. error	.4980	.4980	.4961	.4961
		Max. neg. error	-.4980	-.4980	-.4961	-.4961
		Mean	.0000	.0067	.0000	.0000
		Variance	.0825	.0826	.0816	.0816
10	True rounding	Max. pos. error	.4995	.4995	.4990	.4990
		Max. neg. error	-.4995	-.4995	-.4990	-.4990
		Mean	.0000	.0018	.0000	.0000
		Variance	.0831	.0832	.0829	.0829
	Constant truncation	Max. pos. error	.6647	.6647	.6654	.6654
		Max. neg. error	-.6657	-.6657	-.6654	-.6654
		Mean	-.0553	-.0370	.0000	.0000
		Variance	.0909	.0920	.0937	.0937
	New truncation	Max. pos. error	.4995	.4995	.4990	.4990
		Max. neg. error	-.4995	-.4995	-.4990	-.4990
		Mean	.0000	.0018	.0000	.0000
		Variance	.0831	.0832	.0829	.0829

floating point number system, the dividend is left-shifted when it is less than the divisor to save the iteration for the guard bit. Constant truncation needs to normalize the divisor before division. The new truncation method does not need to normalize the divisor. In the two's complement number system and the negative two's complement number system, constant truncation and the new truncation method can avoid carry propagate addition in the correction step if the divisor is always positive, but the signs of the divisor and the dividend need to be swapped when the divisor is negative.

Table 7.2: Execution comparison of n-bit binary non-restoring division with true rounding, constant truncation and new truncation for unsigned numbers, floating point numbers, two's complement numbers and negative two's complement numbers

Rounding method	Procedure	Number System			
		Unsigned	FLP	2's comp.	Neg. 2's comp.
True rounding	Pre-normalization	no	yes	no	no
	Iterations	n+1	n+1	n	n
	CPA in correction	yes	yes	yes	yes
Constant truncation	Pre-normalization	yes	yes	yes	yes
	Iterations	n	n	n-1	n-1
	CPA in correction	no	no	yes / no	yes / no
New truncation	Pre-normalization	no	yes	no	no
	Iterations	n	n	n-1	n-1
	CPA in correction	no	no	yes / no	yes / no

7.4 Summary

Division is the slowest of the basic arithmetic operations. Traditionally, digit-recurrence division needs an additional iteration of shifting/addition and carry propagate addition for true rounding, increasing the execution time. In this paper, a new truncated division has been proposed and compared to the previous methods showing the same errors to those of true rounding without the additional iteration and carry propagate addition.

Chapter 8

Conclusions

This dissertation presents the algorithms and evaluation of truncated multiplication and truncated division. Since power dissipation is becoming a very important issue, truncated multiplication with nearly correct rounding is attractive for those systems in which a small error is acceptable. The new truncated multiplication has the most symmetric error distribution and smaller maximum errors than the previous methods. The hardware increase is tiny due to the specialized (4:2) counter.

The negative two's complement number system is attractive for signed truncated multiplication to avoid the inherent problem in the conventional two's complement number system. To compare area, delay and dynamic power of the new method to those of the previous methods for the unsigned number system, the conventional two's complement number system and the negative two's complement number system, truncated multipliers were designed in structural HDL, which are automatically generated by perl scripts, and synthesized by Synopsys Design Compiler in standard cells for TSMC 0.25 μm technology. Especially, the dynamic power was estimated by capturing the average switching activity of each cell and net for exhaustive random input patterns. For 8 bit unsigned array truncated multipliers with errors bounded

between -1 and 1 ulp, both previous methods save around 20% of the power consumption compared to the standard full precision multiplier and the new method saves around 30%. For 16 bit negative two's complement array truncated multipliers with errors bounded between -1 and 1 ulp, the previous methods save around 25% of the power consumption compared to the standard full precision multiplier and the new method saves around 30%.

Division is the most complex and the slowest of the basic arithmetic operations. Digit-recurrence division is a widely used class of algorithms for division. This algorithm is performed by iterations of shifts and adds. Each quotient digit is produced by an iteration of a shift and an add. True rounding requires extra hardware and execution time for an additional iteration and carry propagate addition. Even though the previous method for truncated division eliminate these additional costs, it yields more errors than true rounding and needs to have a normalized divisor before division. The new method with a small hardware increase (i.e., one register with size equal to the input word length) achieves errors in the range of (-1/2, 1/2), which are the same as those of true rounding. The benefit of the new method is that it can be performed without additional hardware and execution time to achieve true rounding as well as normalization. This method can be applied to the floating point number system, the conventional two's complement number system and the negative two's complement number system as well as the unsigned number system.

There are many other algorithms to design multipliers and dividers de-

pending on the specific demand. Currently, low power design is hot issue due to the increasing demand of mobile systems and the limitations of battery technology. Even though low power is the current issue, the arithmetic described here is compatible with the demands of mobile applications to process high quality video signals.

Bibliography

[1] Y. C. Lim, "Single-Precision Multiplier with Reduced Circuit Complexity for Signal Processing Applications," *IEEE Transactions on Computers*, Vol. 41, pp. 1333-1336, 1992.

[2] M. J. Schulte and E. E. Swartzlander, Jr., "Truncated Multiplication with Correction Constant," *VLSI Signal Processing, VI*, New York: IEEE Press, pp. 388-396, 1993.

[3] S. S. Kidambi, F. El-Guibaly, and A. Antoniou, "Area-Efficient Multipliers for Digital Signal Processing Applications," *IEEE Transactions on Circuits and Systems, II*, Vol. 43, pp. 90-94, 1996.

[4] E. J. King and E. E. Swartzlander, Jr., "Data-Dependent Truncation Scheme for Parallel Multipliers," *31st Asilomar Conference on Signals, Circuits and Systems*, pp. 1178-1182, 1997.

[5] E. E. Swartzlander, Jr., "The Negative Two's Complement Number System," *Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, In-Press, 2007.

[6] J. E. Thornton, *Design of a Computer: The Control Data 6600*, Glenview, IL: Scott, Foresman and Company, pp. 101-105, 1970.

[7] G. K. Ma and F. J. Tayor, "Multiplier Policies for Digital Signal Processing," *IEEE ASSP Magazine*, Vol. 7, no. 1, pp. 6-20, 1990.

[8] E. E. Swartzlander, Jr., "Truncated Multiplication with Approximate Rounding," *Conference Record of the Thirty-Third Asilomar Conference on Signals, Systems and Computers*, pp. 1480-1483, 1999.

[9] M. J. Schulte, J. E. Stine and J. G. Jansen, "Reduced Power Dissipation Through Truncated Multiplication," *IEEE Alessandro Volta Memorial Workshop on Low Power Design*, Como, Italy, pp. 61-69, 1999.

[10] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*, New York: Oxford University Press, 2000.

[11] M. D. Ercegovac and T. Lang, *Division and Square Root: Digit-Recurrence Algorithms and Their Implementations*, Boston: Kluwer Academic, 1994.

[12] J. E. Robertson, "A New Class of Digital Division Methods," *IRE Transactions on Electronic Computers*, Vol. EC-7, pp. 218-222, 1958.

[13] E. E. Swartzlander, Jr., "High-Speed Computer Arithmetic," Ch. 22 in A. B. Tucker, Jr., Ed., *The Computer Science Handbook*, 2nd Edition, Boca Raton, FL: Chapman & Hall/CRC, pp. 22-15-22-16, 2004.

[14] G. A. Jullien, "High Performance Arithmetic for DSP Systems," in *VLSI Signal Processing Technology*, M. A. Bayoumi and E. E. Swartzlander, Jr. Eds., Boston, MA: Kluwer, pp. 59-96, 1994.

[15] M. D. Ercegovac and T. Lang, *Digital Arithmetic*, San Francisco: Morgan Kaufmann Publishers, pp. 20-25, 2004.

[16] H. L. Garner, "Number Systems and Arithmetic," F. L. Alt and M. Rubinoff, Eds., *Advances in Computers*, vol. 6, New York: Academic Press, pp. 131-194, 1965.

[17] A. D. Booth, "A Signed Binary Multiplication Technique," *Quartely Journal of Mechanics and Applied Mathmetics*, vol. 4, pp. 236-240, 1951.

[18] O. L. MacSorley, "High-Speed Arithmetic in Binary Computers," *IRE Proceedings*, vol. 49, pp. 67-91, 1961.

[19] M. J. Flynn, "On Division by Functional Iteration," *IEEE Transactions on Computers*, Vol. C-19, pp. 702-706, 1970.

[20] S. F. Anderson, J. G. Earle, R. E. Goldschmidt and D. M. Powers, "The IBM System/360 Model 91: Floating-Point Executionm Unit," *IBM Journal of Research and Development*, Vol. 11, pp. 34-53, 1967.

[21] L. Dadda, "Some Schemes for Parallel Multiplier," *Alta Frequenza*, Vol. 34, pp. 349-356, 1965.

[22] C. S. Wallace, "A Suggestion for a Fast Multiplier," *IEEE Transactions on Electronic Computers*, Vol. 13, pp. 14-17, 1964.

[23] K. Bickerstaff, M. J. Schulte, and E. E. Swartzlander, Jr., "Parallel Reduced Area Multipliers," *Journal of VLSI Signal Processing*, Vol. 9, pp. 181-191, 1995.

[24] S. S. Mahant-Shetti, P. T. Balsara, and C. Lemonds, "High Performance Low Power Array Multiplier Using Temporal Tiling," *IEEE Transactions on VLSI Systems*, Vol. 7, pp. 121-124, 1999.

[25] S. D. Pezaris, "A 40-ns 17-Bit by 17-Bit Array Multiplier," *IEEE Transactions on Computers*, Vol. C-20, pp. 442-447, 1971.

[26] K. Hwang, "Global and Modular Two's Complement Cellular Array Multipliers," *IEEE Transactions on Computers*, Vol. C-28, pp. 300-306, 1979.

[27] C. Baugh and B. Wooley, "A Two's Complement Parallel Array Multiplication Algorithm," *IEEE Transactions on Computers*, vol. C-22, pp. 1045-1047, 1973.

[28] T. P. Ryan, *Statistical Methods for Quality Improvement*, New York: John Wiley & Sons, 2000.

[29] R. Pratab, *MATLAB: A Quick Introduction for Scientists and Engineers*, 2nd Edition, New York: Oxford University Press, 2002.

[30] A. Gilat, *MATLAB: An Introduction with Applications*, 2nd Edition, Hoboken, NJ: John Wiley & Sons, 2004.

[31] B. W. Kernighan and D. M. Ritchie, *The C Programming Language*, 2nd Edition, Murray Hill, NJ: Prentice Hall, 1988.

[32] B. Stroustrup, *The C++ Programming Language*, 3rd Edition, Murray Hill, NJ: Addison Wesley, 1997.

[33] M. J. Schulte and K.E. Wires, "High-Speed Inverse Square Roots," *Proceedings of the 14th IEEE Symposium on Computer Arithmetic*, pp. 124-131, 1999.

[34] D. E. Thomas and P. R. Moorby, *The Verilog Hardware Description Language*, 5nd Edition, Norwell, MA: Springer, 2002.

[35] S. Palnitkar, *Verilog HDL*, 2nd Edition, Sunnyvale, CA: Prentice Hall, 2003.

[36] R. L. Schwartz, *Learning Perl*, 2nd Edition, Sebastopol, CA: O'Reilly Media, 1997.

[37] L. Wall, T. Christiansen and J. Orwant, *Programming Perl*, 3rd Edition, Beijing: O'Reilly Media, 2000.

[38] *Power Complier: User Guide*, Synopsys Corrporation, 2003.

[39] J. B. Sulistyo and D. S. Ha, "A New Characterization Method for Delay and Power Dissipation of Standard Library Cells," *VLSI Design*, Vol. 15, pp. 667-678, 2002.

[40] J. B. Sulistyo, J. Perry, and D. S. Ha, *Developing Standard Cells for TSMC 0.25um Technology under MOSIS DEEP Rules,* Department of Electrical and Computer Engineering, Virginia Tech, Technical Report VISC-2003-01, 2003.

[41] J. T. Coonen, "An Implementation Guide to a Proposed Standard for Floating-Point Arithmetic," *IEEE Computer*, Vol. 13, pp. 68-79, 1980.

[42] American National Standards Institute and Institute of Electrical and Electronic Engineers, *IEEE Standard for Binary Floating-Point Arithmetic,* ANSI/IEEE Standard, Std. 754-1985, 1985

Vita

Hyuk Park was born in Pusan, Korea on May 16, 1973, the son of Yong Gon Park, and Kyeong Ja Kim. After completing his work at Hyekwang High School, Pusan, Korea in 1992, he entered the Hanyang University, Ansan, Korea in 1992. He received the degree of Bachelor of Science from the Hanyang University in February of 1999. In September of 2000, he entered the Graduate School at the University of Texas. In 2001, he joined Application Specific Processor Group at the University of Texas at Austin. In May of 2002, he received the degree of Master of Science in Engineering at the University of Texas at Austin. Since then, he has been pursuing his Doctorate in Engineering at the University of Texas at Austin. During his graduate studies, he worked at Xilinx Incorporated and Samsung Electronics.

Permanent address: 1644 W 6th st. #D
Austin, Texas 78703

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.